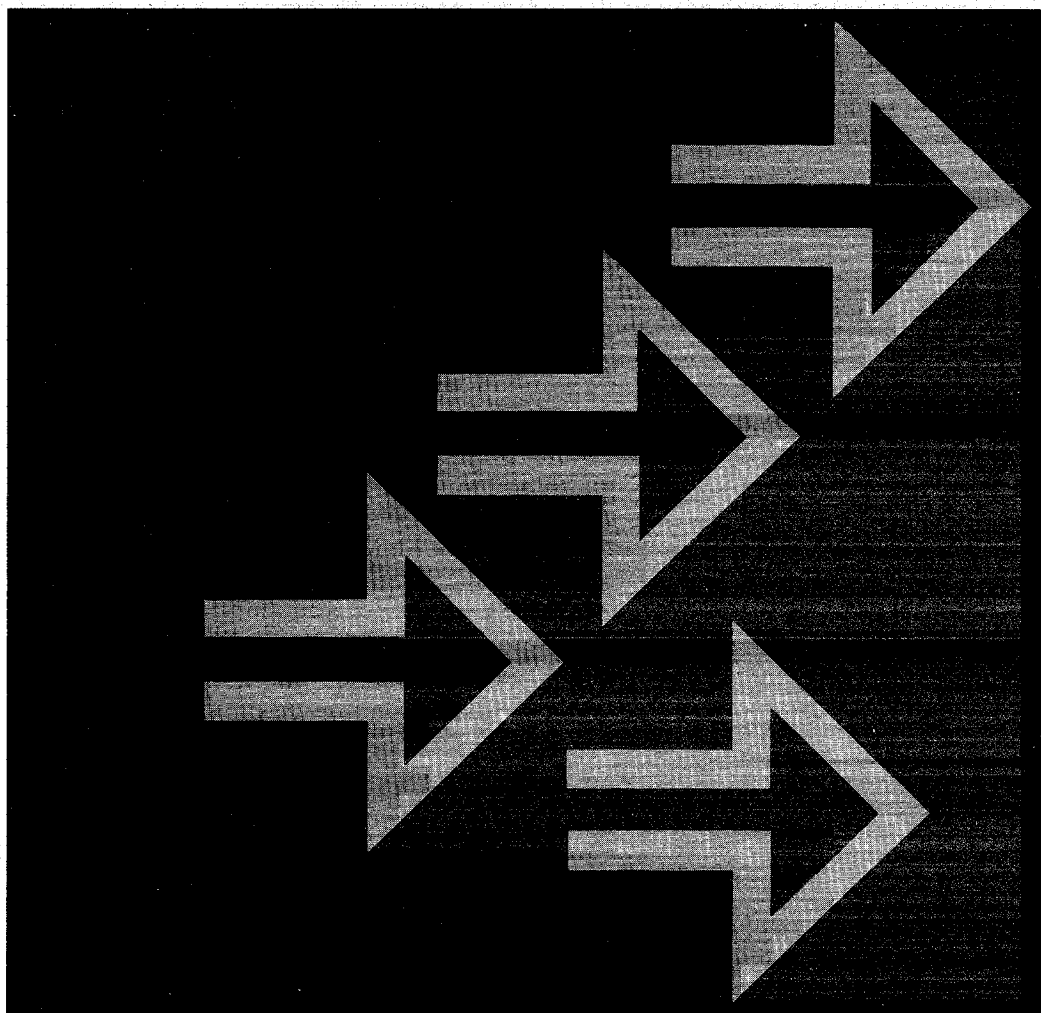


**Document Composition Facility
Generalized Markup Language**

SH20-9187-06

Starter Set Reference

Release 4.0



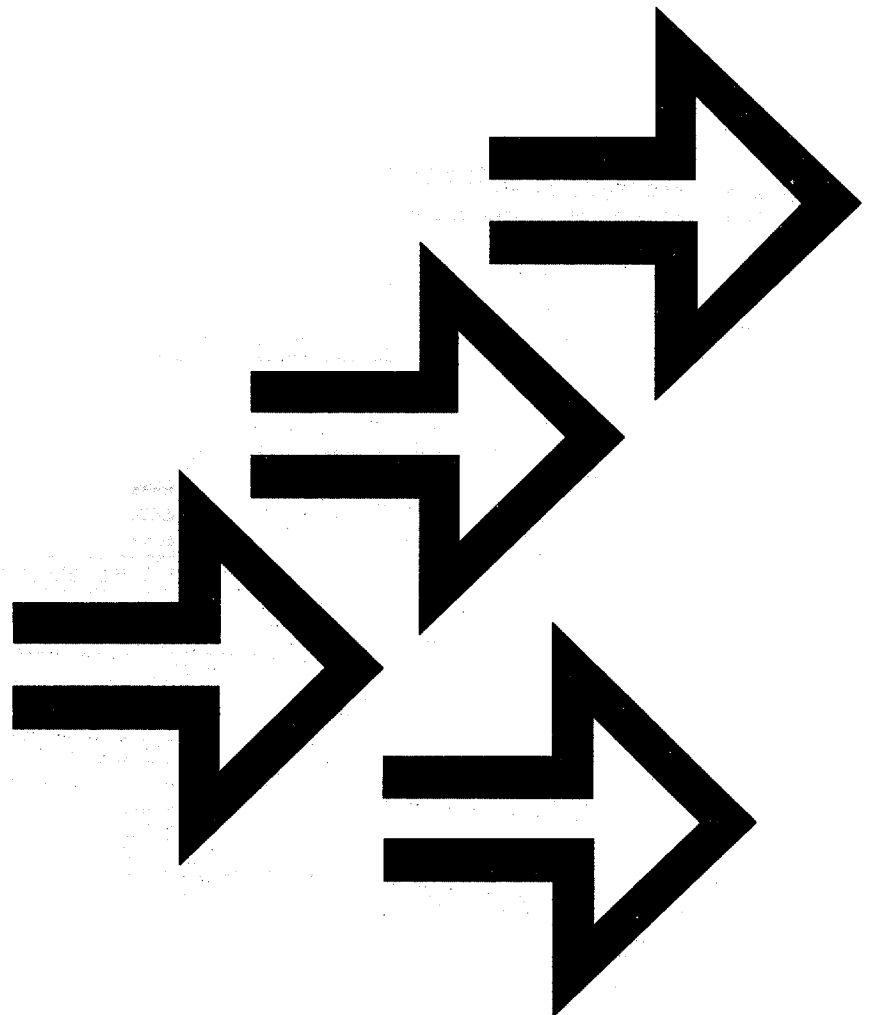


Document Composition Facility Generalized Markup Language

SH20-9187-06

Starter Set Reference

Release 4.0



This book was formatted with IBM Publishing Systems BookMaster licensed program (Program Number 5688-015), using the *sslike* style file. When you use the GML starter set tags, your formatted output may differ from the examples shown in this book because of differences in column line length, fonts, line spacing, device types, indention, and other formatting differences between GML and BookMaster.

Note!

Before using this information and the product it supports, be sure to read the general information in "Notices" on page xi.

Seventh Edition (January 1991)

This edition applies to Release 4.0 of the Document Composition Facility licensed program, Program Number 5748-XX9, and to all subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Be sure to use the correct edition for the level of the product.

Technical changes in this edition are marked by vertical bars in the left margin.

Consult the latest edition of the applicable IBM system bibliography for current information on this product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation
Information Development
Department 588
P.O. Box 1900
Boulder, Colorado 80301-9191

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1980, 1991. All rights reserved.
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Introduction	1
How to Use This Publication	2
Phrases Used in This Book	3
Using GML	4
Objectives of GML Markup	5
Advantages of Using SCRIPT/VS and GML	5
GML with Other Applications	7
How SCRIPT/VS Interprets GML	7
 Chapter 2. Marking Up General Documents	 11
Markup Concepts	11
Elements in a General Document	15
Lists	16
Definition Lists	17
Examples	17
Glossary Lists	17
Figures	18
Tables	18
Footnotes	18
Highlighted Phrases	19
Cross-References	19
Index	20
Entering GML Markup	20
Entering Text	22
SCRIPT/VS Symbols	23
Control Words and Macros	26
Source Document Management	26
 Chapter 3. GML Starter Set Tags	 35
:ABSTRACT—Document Abstract	37
:ADDRESS—Address	38
:ALINE—Address Line	38
:APPENDIX—Appendix Section	39
:BACKM—Back Matter Section	41
:BODY—Body Section	42
:CIT—Title Citation	43
:DL—Definition List	44
:DTHD—Definition Term Heading	44
:DDHD—Definition Description Heading	45
:DT—Definition Term	45
:DD—Definition Description	45
:LP—List Part	45
:FIG—Figure	48
:FIGCAP—Figure Caption	48
:FIGDESC—Figure Description	48
:FIGLIST—List of Illustrations	52
:FIGREF—Figure Reference	53
:FN—Footnote	54

:FNREF—Footnote Reference	56
:FRONTM—Front Matter Section	57
:GDOC — General Document	58
:GL—Glossary List	60
:GT—Glossary Term	60
:GD—Glossary Definition	60
:LP—List Part	60
:H0 — :H6—Headings	62
:HDREF—Heading Reference	65
:HP0 — :HP3—Highlighted Phrases	66
:I1, :I2, and :I3—Index Entries	68
:IH1, :IH2, and :IH3—Index Headers	71
:INDEX—Index	74
:IREF—Index Entry Reference	75
:LIREF—List Item Reference	77
:SL, :OL, :UL—Lists	78
:LI—List Item	78
:LP—List Part	78
:LQ—Long Quotation	80
:NOTE—Note	81
:P and :PC—Paragraph	82
:PREFACE—Preface	84
:PSC—Process-Specific Control	85
:Q—Quoted Phrase	87
:RDEF—Row Definition	88
:TABLE—Table	93
:ROW—Row	94
:C—Cell	95
:TFT—Table Footer	96
:THD—Table Header	97
:TCAP—Table Caption	98
:TDESC—Table Description	98
Examples of Tables	99
:TLIST—List of Tables	108
:TNOTE—Table Note	109
:TREF—Table Reference	110
:TITLEP—Title Page	111
:TITLE—Document Title	111
:DOCNUM—Document Number	111
:DATE—Document Date	111
:AUTHOR's—Author Name	111
:TOC—Table of Contents	114
:XMP—Example	115
 Chapter 4. Processing GML Documents with SCRIPT/VS	 117
SCRIPT Command Options	117
Processing Options	118
Logical Device and Output Destination	124
Error Handling	126
Font Requirements for Page Printers	126
Font Requirements for PostScript Devices	127
Overriding the Default Font Using the CHARS Option	127
Issuing the SCRIPT Command	128
SCRIPT/VS Logical Devices	128
SCRIPT/VS Logical Line Devices	128
SCRIPT/VS Logical PostScript Devices	129
SCRIPT/VS Logical Page Devices	129
SCRIPT/VS Logical Page Devices (cont.)	130
CMS Environment	130
TSO Environment	131

ATMS-III System	132
Batch Processing	133
SYSVAR Option	133
Interim Processing	136
Markup Errors	137
Appendix A. The DSMPROF4 Profile	139
GML Tag-to-APF Mapping	139
Formatting Parameters	139
Epifile	144
Appendix B. The DSMGML4 Macro Library	145
Naming Conventions	145
Application Processing Functions	145
Major Document Parts	145
Citations	146
Examples	146
Figures	146
List of Illustrations	147
Footnotes	147
Headings	147
Highlighted Phrases	147
Lists	147
Indexing	148
Long Quotations	148
Paragraphs	148
Row Definitions	148
Tables	149
List of Tables	150
Process-Specific Controls	150
Quotations	150
Title Page	150
General Processing	150
Identifiers	150
SYSVAR R and W	151
Running Headings and Footings	152
Messages and Text Constants	152
Imbed Tracing	152
Cross-Reference Listing	152
Appendix C. Compatibility with Earlier Releases of SCRIPT/VS	153
Release 4.0 of the Document Composition Facility	153
Appendix D. Related Publications and Products	157
Related Products	157
Optional Features	157
Publication Library Guide for the Document Composition Facility	158
DCF Publications	159
Related Publications	160
Glossary	161
Source Identifiers	161
References	161
Index	175

Figures

1.	Processing Documents with GML	8
2.	Different Tags Processed by the Same APF	8
3.	Multiple Profiles for Different Applications	9
4.	Contents of a Paragraph Unit	12
5.	Basic Document Elements	13
6.	Structure of a General Document	14
7.	Miscellaneous Elements	15
8.	Contents of a Table	18
9.	Markup Example	25
10.	The Title Page of a Document	113
11.	Symbols and Initial Formatting Parameters Set in DSMPROF4	140
12.	Item Identifiers for Ordered and Unordered Lists	143

Tables

1.	Space-Units Notation	22
2.	Summary of Head-Level Tags	63
3.	Example of a Column-Wide Simple Table	99
4.	Example of a Page-Wide Simple Table	100
5.	A More Complex Table	101
6.	Place of Offense and Residence of Offender	106
7.	SCRIPT/VS Logical Line Devices	128
8.	SCRIPT/VS Logical PostScript Devices	129
9.	SCRIPT/VS Logical Page Devices	129
10.	SCRIPT/VS Logical Page Devices (cont.)	130
11.	Items with Name Changes for DCF Release 4.0	154

Summary of Amendments

Release 4.0

Document Composition Facility (DCF), Release 4.0, includes the following:¹

- **Shading:** Allows for shading of boxes, areas, and table cells on Advanced Function Printing (AFP) printers and PostScript devices.
- **MVS and VM Extended Architecture Exploitation:** Allows the DCF program to run and use storage above 16 megabytes.
- **Online Help:** Online help for the SCRIPT command is now available for the VM/CMS and MVS/TSO systems.
- **Online Help for Error Messages:** Online GML and SCRIPT error messages are now available for VM/CMS and MVS/TSO systems.
- **Separation Masters:** Allows construction of masters for multiple color printing and multi-part forms.
- **Including Overlays:** Allows you to merge page overlays in a DCF document that is formatted for an AFP printer.
- **Page Segment Enhancements:** DCF allows page segments containing Image Object Content Architecture (IOCA) Image Objects to be included when formatting for some AFP page printers.

The new ABSOLUTE parameter for the .SI [Segment Include] control word allows the user to specify the depth and width of the space to be reserved on the .SI control word instead of using the size of the segment.

- **File Size Limit:** Removes the VM restriction for the number of records in an input file of 65,535.
- **PRINT Option:** Allows for use of the PRINT option on the SCRIPT command in VM. Files sent to AFP page printers² will go directly to SPOOL saving processing time and DASD.
- **LaserPrinter 4028 Support:** Adds support for the LaserPrinter 4028.
- **Font Character Set Size:** Raises the limit on the number of characters allowed in a font character set from 292 to approximately 1000.

¹ Release 4.0 of the Document Composition Facility may be used with Release 3 of the Document Library Facility.

² The PRINT option is still ignored for the IBM 4250 printer and PostScript devices.

- **Language Attribute on the :GDOC tag:** Allows for the selection of the language of the document, which includes translations for literals and messages, and using the correct language for spelling verification, hyphenation, and index sorting sequences in 15 different languages when using the starter set.
- **Hyphenation:** Adds six new languages for spelling verification and hyphenation, and algorithmic hyphenators for all 15 languages.
- **Revision Code Font:** Allows for selecting the fonts used in revision code characters.
- **Revision Character Alignment:** Allows for specifying revision codes on the left or the right or on alternate columns. For a two column format, the revision code character can appear to the left of the first column and to the right of the second column.
- **Table Storage Relief:** The new BREAK parameter for the .TA [Table] control word allows the user to break a table, which places the portion of the table that has been formatted, and releases the storage used for formatting that portion.
- **Page Number Symbol Limit:** Allows for the page numbering limit to increase to 99,999,999.
- **Output Comment:** Allows for use of the .OC [Output Comment] control word with AFP printers. The .OC control word is still ignored when formatting for the 4250 printer.
- **Schedule Tag Set:** Provides a set of tags to allow printing of schedules with start dates, end dates, projected dates, actual dates, slip dates, actions, responsible parties, and schedule changes.
- **Overhead Transparency Tag Set:** Provides a set of tags to allow printing of overhead transparencies with running headers, running footers, and a large, easy to read font.
- **Memo Tag Set:** Provides a set of tags to allow printing of memos in a consistent style.
- **TSO Changes to DCF:** Changes have been made to DCF to improve usability in the TSO environment. The changes can be summarized as follows:
 - Addition of TSO command options
 - Addition of TSO DDNAMES for SCRIPT command files
 - Changes to the Font Library Index Program (FLIP) to allow concatenated font libraries.
- **Module Renaming:** Allows a VM system programmer to change the names used for the DCF disk-resident module, any discontinuous shared segment, and the bootstrap module at installation time.
- **Font Library Index Program (FLIP) Enhancement:** In MVS/TSO, FLIP will search all of the data sets that have been designated as part of a concatenated font library. In VM/CMS, FLIP searches disk extensions for fonts of the specified filetype.

Notices

References in this publication to products or services of IBM do not imply that IBM will make them available in all countries where IBM does business or that only products or services of IBM may be used. Noninfringing equivalents may be substituted, but the user must verify that such substitutes, unless expressly designated by IBM, work correctly. No license, expressed or implied, to patents or copyrights of IBM is granted by furnishing this document.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries:

IBM®	Advanced Function Printing™
PSF™	Print Services Facility™
MVS™	TSO™
VM™	VSE™
BookMaster®	ProcessMaster®
BookManager™	DisplayWrite®
VM/XA™	Virtual Machine/Extended Architecture™
ATMS-III™	IBM 3800 Printing Subsystem Models 3 and 6™
CICS™	IBM 3900 Advanced Function Printer™
AFP™	IBM 3825 Page Printer™
IBM 3827 Page Printer™	IBM 3828 Advanced Function MICR Printer™
IBM 3835 Page Printer™	IBM 3812 Page Printer™
IBM 3816 Page Printer™	IBM LaserPrinter 4028™
IBM 4224-2xx Page Printer™	IBM 4234 Page Printer™
IBM 3820 Page Printer™	IBM LaserPrinter 4019™
IBM 4250 Printer™	

The following terms are trademarks of other companies as follows:

PostScript	Adobe Systems Incorporated
Monotype	Monotype Corporation, plc.
Monotype Times New Roman	Monotype Corporation, plc.
Times Roman	Allied Linotype or its subsidiaries
New Century Schoolbook	American Type Foundry

The fonts provided by IBM are frequently designed by graphic designers in other companies and adapted for IBM printers. The IBM font listed below is a functional equivalent of a font developed by another company:

IBM Font	Functional Equivalent
Sonoran Serif	Monotype Times New Roman

Central Programming Service support and maintenance are available for the following Generalized Markup Language (GML) profiles and macros:

- The GML starter set profile: DSMPROF4
- The GML schedule profile: DSMSPROF
- The GML overhead transparency profile: DSMTPROF
- The GML memo profile: DSMMPROF
- The GML bar-code profile: DSMBPROF
- The GML SCRIPT Mathematical Formula Formatter profile: DSMFPROF
- The GML office document feature profile: DSMOPROF
- The GML office document feature macro-instruction library: DSMOGML
- The GML SCRIPT Mathematical Formula Formatter macro-instruction library: DSMFMAC
- The GML macro-instruction library: DSMGML4

Support and maintenance, however, *are not* available for these profiles and macros if they have been modified in any way. If you modify these items, it is recommended that you also maintain an *unmodified* version of these items for diagnostic purposes.

An authorized program analysis report (APAR) may not be submitted regarding the translation of literals, messages, and phrases in the starter set.

Chapter 1. Introduction

The Generalized Markup Language (GML) is a language that describes the *parts* of a document. GML provides the syntax and usage rules for using descriptive *tags*, without concerning yourself with how the document will be processed.

IBM provides an implementation of GML, written in the SCRIPT/VS formatting language, as an example with the Document Composition Facility. This manual describes that GML *starter set* of tags and the formatting results that SCRIPT/VS produces when these GML tags are used. It provides basic information on using SCRIPT/VS to process GML documents and explains how to modify the starter set for your organization's requirements. Modifying or adding tags requires a knowledge of SCRIPT/VS control words, symbols, and macros, as well as knowledge of the GML starter set.

This publication is intended as a reference for those people who are using the GML starter set for their document preparation and for those who will be modifying the starter set tags and creating new tags for their location's specific needs. The *Document Composition Facility: Generalized Markup Language Starter Set Implementation Guide* describes the Generalized Markup Language starter set profile and macro library and presents detailed information for modifying starter set tags.

References to the 3800 Printing Subsystem apply to the 3800 Printing Subsystem Model 1 and the 3800 Printing Subsystem Models 3 and 6 (in compatibility mode), unless otherwise explicitly stated.

References to page printers apply to the following unless otherwise explicitly stated:

- AFP page printers
 - 3800 Printing Subsystem Models 3 and 6
 - 3812 PagePrinter
 - 3816 Page Printer
 - 3820 Page Printer
 - 3825 Page Printer
 - 3827 Page Printer
 - 3828 Advanced Function MICR Printer
 - 3835 Page Printer
 - 3900 Advanced Function Printer.
 - LaserPrinter 4028
 - 4224 Printer
 - 4234 Printer
- 4250 Printer
- PostScript devices configured to accept 8-bit ASCII

The IBM 4250 Printer is an electroerosion printer that produces camera-ready copy. The Composed Document Print Facility (CDPF) licensed program is required to print DCF formatted output on the 4250.

Note: Although you can use DCF to format for a 4250 printer, the 4250 Printer and CDPF are no longer available from or supported by IBM.

When you use the GML starter set tags, your formatted output may differ from the examples shown in this book because of differences in column line length, fonts, line spacing, device types, indentation, and other formatting differences between GML and BookMaster.

How to Use This Publication

This document is divided into the following sections:

- Chapter 1, "Introduction" describes the concept of a generalized markup language and compares it to traditional methods of document markup. The advantages of GML are presented.
- Chapter 2, "Marking Up General Documents" describes the structure of general documents and how the parts, or *elements*, of the document can be identified with *tags*. Guidelines for entering text and markup in documents to be processed with GML are suggested.
- Chapter 3, "GML Starter Set Tags" describes each of the tags provided with the GML starter set.
- Chapter 4, "Processing GML Documents with SCRIPT/VS" describes the SCRIPT command and the options of the SCRIPT command for processing documents marked up in GML. The aspects of formatting that can be controlled through the use of SCRIPT/VS system variables (SYSVARs) are described.
- Appendix A, "The DSMPROF4 Profile" outlines the contents of the file DSMPROF4, which is the GML starter set profile for Release 4. The aspects of formatting that can be varied under control of parameters set in DSMPROF4 are described.
- Appendix B, "The DSMGML4 Macro Library" outlines the contents of the DSMGML4 macro library, which contains the APFs for the GML starter set.
- Appendix C, "Compatibility with Earlier Releases of SCRIPT/VS" contains information on using DSMPROF4 with earlier releases of SCRIPT/VS.
- Appendix D, "Related Publications and Products" lists the DCF publications as they relate to user tasks, and related products.

Phrases Used in This Book

Several descriptive phrases are used throughout this book. An introduction to them here will help you understand them when you come upon them later—or you can refer to them as needed. A glossary at the back of this book contains an extensive listing of words and phrases used.

Element	Any part of a document: a single character, a word, or a sentence. It also refers to any part of a document you can identify with a GML tag (tagged element), such as paragraphs, figures, or headings.
Paragraph unit	Paragraphs and other document elements that have the same structure as a paragraph, such as notes and paragraph continuations.
Implied paragraph	Paragraphs for which no :P tag has been entered, such as definition descriptions, list parts, figure descriptions, footnotes, list items, and long quotations.
Text item	Tagged elements that can occur within a paragraph unit, such as title citations, references, highlighted phrases, and quotation marks.
Single text line	A line of text that must appear entirely on the same line as its tag or entirely on the line following its tag, such as definition terms, figure captions, and headings. The single text line cannot contain any tags.
Line element	A line of text that appears in the output exactly as it does in the input document. Line elements appear in figures and examples.
Basic document elements	The tagged elements that occur frequently in a general document, and have text associated with the tag, such as paragraphs, headings, and footnotes.
Basic document structures	The tagged elements that indicate the basic structure of a general document, such as the front matter, the body, and the various lists. These elements don't have any text specifically associated with the tag.

Using GML

You may be accustomed to typing documents on typewriters, where they appear exactly the way you typed them. Or, you may have worked with word processing or text-processing systems where you *marked up* the text by entering special codes to achieve a desired output appearance. In both cases, you had to be concerned with such things as the length of lines and pages, hyphenation of words, alignment of columns, indentions, tab settings, and similar formatting considerations.

Markup with GML is different—and much easier. You look over a document to determine its parts, or *elements*, such as paragraphs, figures, lists, and headings. Then you insert *element tags* to show where elements begin and end, and what type of element it is. For example, :FIG indicates the start of a figure and :EFIG indicates the end of that figure.

Not all elements require an ending tag. The tag :TITLE starts the title of the document; the title ends at the end of the input line. (The individual tag descriptions, presented later in this manual, indicate when an ending tag is required.)

You can also insert *attributes* with tags to give other information about the element, such as the type of *frame* to be used in setting off a figure or the *ID* of a heading. For example:

```
:fig frame=box.
```

```
:  
:efig.
```

or

```
:h1 id=gml.GML Markup Guide
```

The document into which you enter the text and GML markup is called the *source document*. It is created using the facilities for text entry and editing available at your installation. You may, for example, be using the ISPF/PDF editor in TSO or XEDIT in CMS.

You do not have to put formatting codes into the source document because a text programmer has already created processing instructions for each element that you identify with the GML tags. These instructions are called *application processing functions* (APFs). For example:

- The APF for a simple list (which you indicate by specifying the GML tag :SL) causes the text to be indented
- The APF for an item (:LI) of an ordered list (:OL) causes the item to be numbered automatically.

The formatting instructions for SCRIPT/VS (a text-processing program), called *control words*, are in the APFs that get associated with the tags you use. The control words are not in your source document. That is why you do not have to be concerned with formatting when you mark up documents with GML.

Once the source document is prepared, you call *SCRIPT/VS* to process it by using the SCRIPT command. SCRIPT/VS reads and interprets all the tags and produces a formatted *output document*.

Because of GML, you can have the output document formatted in several different ways without changing your markup. When you call SCRIPT/VS to process your document, you specify a file that associates each of the tags in your document with a specific APF. (This file is called a *profile* and has also been prepared by a text programmer.) SCRIPT/VS will then perform all of the hyphenation, justification, page numbering, page layout, spelling verification, table of contents and index preparation, and other processing that might be required.

Objectives of GML Markup

SCRIPT/VS functions are invoked by SCRIPT/VS interpreting the *markup* in your source document. To mark up a source document is to add information to it that enables a person or system to process it in some way. In SCRIPT/VS, the added information, or markup, can be control words, or it can be GML tags that describe the characteristics of a source document without respect to particular processing.

Although some SCRIPT/VS control words are used for general document handling and for representing information with symbols, the control words we usually think of when we talk about document markup are the ones that control the actual formatting of the document. For example, control words would allow you to specify that text should be set in a column 25 picas wide. These formatting control words would always have the same meaning, no matter what the content of that column was.

GML markup, on the other hand, has an entirely different purpose. GML is a descriptive language—with GML you describe the parts of a document. For example, GML markup allows you to identify that certain text is a paragraph, or that other text is part of a numbered list. That paragraph will always be a paragraph in this document, independent of how it is formatted. The authors of documents can specify this kind of markup without understanding the composition or publishing conventions that will be used. They can concentrate entirely on the content and purpose of the document, which is their own field of expertise.

When a source document is marked up with GML, it can be processed in various ways, merely by providing different interpretations for the same set of tags through the use of different profiles. These interpretations can result in different composition, or they can even result in processing other than formatting.

Advantages of Using SCRIPT/VS and GML

Some benefits in using GML when you use SCRIPT/VS for text processing are:

- **Alternative GML interpretation.** A GML tag need not be limited to a single SCRIPT/VS interpretation. For example, a tag might indicate that a group of words in the text is a paragraph. For one application, you might want the first line of a paragraph to be indented. But for another application, you might want paragraphs to be set in block style with spaces between them. Each application can be satisfied by alternative GML interpretations, *with no change* to the source document or to the markup of paragraphs. You have control over the way GML is interpreted by the profile you specify.

- Ease of markup. It's easy to remember GML tags because they consist of terms and abbreviations commonly used to describe a document. GML generally requires fewer characters to be entered for markup than a corresponding complex sequence of control words. The result is faster markup and keying of the document. Changes to the markup are also faster and might even be eliminated with GML because you can control how GML is interpreted by the profile you specify.
- Ease of text update. It's easy to update text marked up with GML. With GML, such things as the numbering of items in an ordered list are usually left to the formatter, which numbers the items automatically. Thus, when you insert or delete an item, you don't have to renumber the following items, as you would if the numbers were part of the source text.
- Uniformity of formatting style. Use of GML for all documents of a particular kind results in a single format for all those documents without the people who do markup even having to think about format. Similarly, a change in format for all the documents could be achieved simply by changing the way GML is interpreted without anyone having to be retrained to do markup in a different way.
- Document exchange. Documents with GML can be processed by different groups or locations more easily than documents with the specific markup of a particular *house style*. With GML, you can obtain your own formatting style and support your own output devices by using your own interpretation of the tags.

Additional benefits result from the automatic processing that SCRIPT/VS does for you when you use GML.

- The running footing is generated from the level 0 and level 1 headings, the title page, or both. The text of the running footing depends on whether you are printing the document for duplexing and whether you are using a title page.
- Pages are numbered automatically, with the pages in the front matter printed in lowercase Roman numerals and the pages in the body and back matter printed in Arabic numerals.
- Figures with figure captions are automatically numbered sequentially. The figure numbers and captions are saved and printed in a "List of Illustrations" where requested with the :FIGLIST tag.
- Level 0 through level 4 headings located in the body of the document are automatically sequentially numbered if so requested.
- A table of contents and index are automatically generated and included where you specify the :TOC and :INDEX tags.
- Cross-references can be made to headings, footnotes, figures, tables, list items, and index entries. Both the FPASSES and TWOPASS options of the SCRIPT command, and in CMS and TSO, the SYSVAR R and W of the SCRIPT command, allow you to make forward references.

GML with Other Applications

For applications involving general processing with SCRIPT/VS or other programs the benefits of GML include:

- Alternative text-processing programs. SCRIPT/VS could be used to interpret GML into the processing controls of text-processing programs other than SCRIPT/VS. For example, they might be interpreted into the controls of a composition program (such as the IBM Installed User Program TERMTEXT/Format, 5796-PBR), which could be used to produce output for a photocomposition device.
- Database applications. GML describes the contents of documents so that programs can identify information in them. Programs could be written alone or in conjunction with SCRIPT/VS processing to extract information from documents for database construction or to retrieve information for database sharing.
- Unique customer applications. Precisely because GML is general, it allows for numerous applications unique for a particular customer. GML designed for this purpose could, for instance, be used both for formatting printed output and for retrieving information.

How SCRIPT/VS Interprets GML

In GML, *interpreting* a tag means performing the correct processing function on its associated text. The choice of processing functions will depend upon the desired application. For example, obtaining a proof copy of a document, where you might want double-spaced lines and wide margins, would be considered a different application from obtaining final copies for distribution. In each case, different processing functions for some of the tags would be used. The tags would not change—only the associated APFs.

In SCRIPT/VS, APFs are sets of control words. These sets of control words are not usually placed in the source document itself. They can be placed in a separate document called a *profile*. SCRIPT/VS processes the profile before it processes the document marked up in GML. Profiles for different kinds of applications can have different APFs for some tags and common APFs for others.

SCRIPT/VS has still greater flexibility in that each APF can also be defined as an individual document. These APFs are defined in documents separate from the profile and are stored in host system libraries, so they will be available for the processing of many documents and applications. In the CMS environment, APFs are stored in a macro library. In TSO they are located in a partitioned data set (PDS). In ATMS-III, APFs are stored in SYSOP (system operator) permanent storage. Figure 1 on page 8 shows a source document, a profile, and three APFs established as separate documents within the host system library.

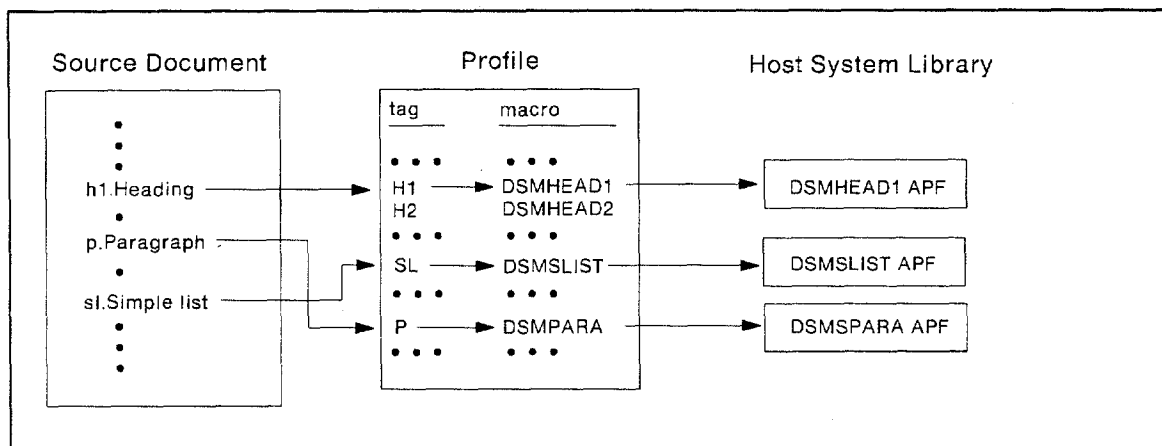


Figure 1. Processing Documents with GML. The profile provides the mapping between the tags you have used in your source document, which identify elements of text, and APFs located in the host system library, which provide formatting functions.

You may want a single APF to operate on the text associated with many different tags, or a single tag's text to be operated on by a number of different APFs for different applications. To accomplish this, SCRIPT/VS provides a way of associating a tag with the name of the APF that is to act on its text. As SCRIPT/VS processes a document marked up with GML, it interprets GML tags by using the APFs. SCRIPT/VS is directed from the tag to the APF through the tag-to-APF association within the profile you specify. If SCRIPT/VS encounters a tag that has not been associated with an APF, it attempts to find a macro or control word with the same name as the tag.

One or more tags can be associated with the same APF. This will cause the same processing to occur for the text associated with each tag. Figure 2 shows this relationship.

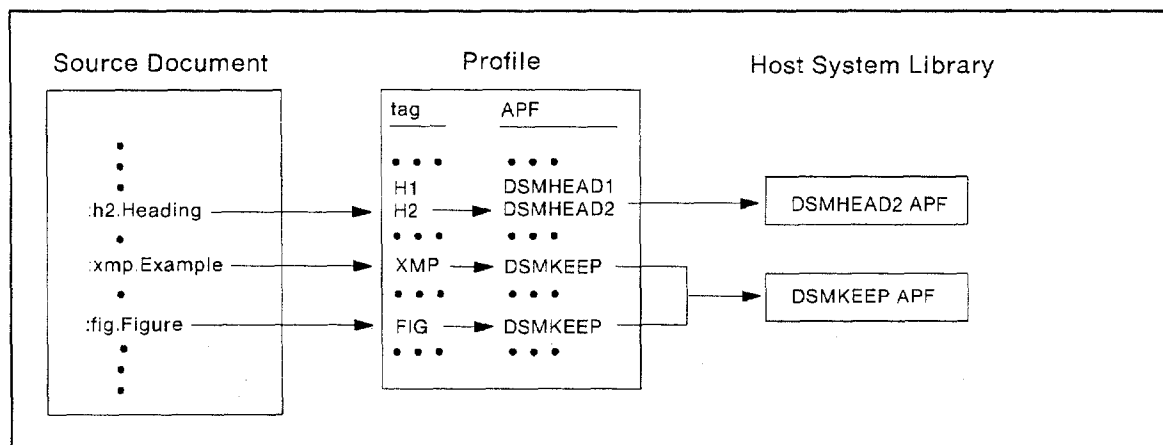


Figure 2. Different Tags Processed by the Same APF. A single APF can provide formatting functions for several tags.

Figure 3 illustrates the use of multiple profiles. When you run SCRIPT/VS, you indicate the profile for the application being run. The profile then associates each tag with the correct APF for that application.

SCRIPT/VS provides the facilities for your organization to define the profiles and APFs as you want them. Profiles and APFs are just documents with special uses. You create them, update them, and store them as you do your text documents.

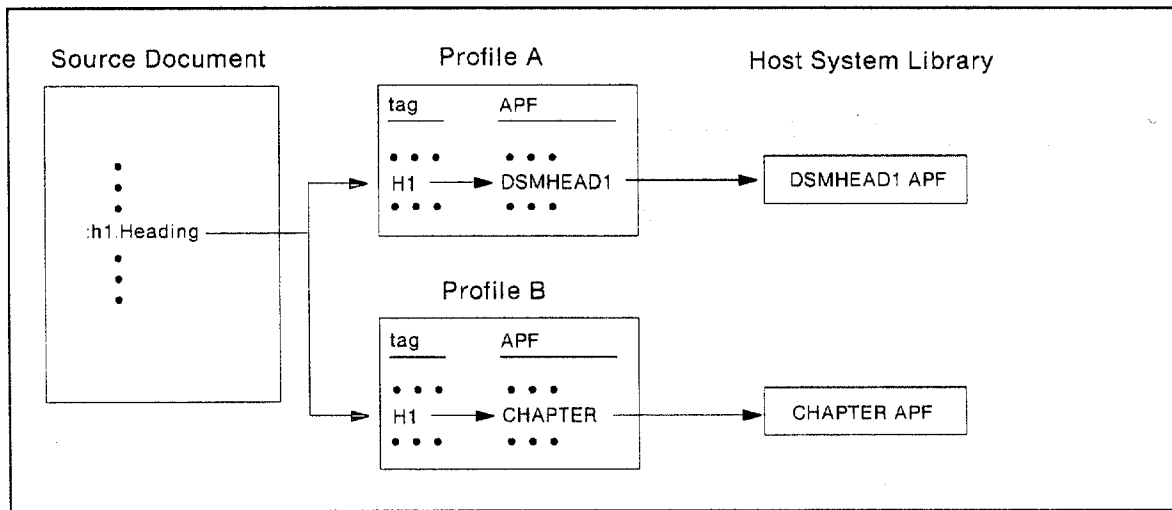


Figure 3. Multiple Profiles for Different Applications. A single document can be used for different purposes by specifying different profiles. The tags in the source document remain unchanged.

Chapter 2. Marking Up General Documents

When you mark up a document, you must know what type of document it is, so that you can use the correct markup procedures for that type. For example, if your installation produces specialized types of documents, such as directories, procedure manuals, or standard contracts, separate sets of tags and markup practices might have been created for each of them. If your document does not fit into a specialized category, it might be a *general document*. The common paper size for general documents is 8-1/2 by 11 inches. This publication describes the tags and markup practices for general documents provided with the starter set.

Markup Concepts

Documents are not just arbitrary strings of characters or words; they have a natural structure, which in some cases can be fairly elaborate. This section explains the structure of general documents so you will know how to mark them up.

The smallest things in a document are individual characters. Characters are combined to form words. Groups of words are phrases, which are combined to form sentences. Sentences themselves are elements of still larger structures, such as paragraphs.

Paragraphs, and other document elements that have the same structure as a paragraph, are called *paragraph units*. Most of the time, elements that occur within a paragraph unit can be identified clearly in the text, and you will rarely need to mark them up with tags. For example, the bounds of a word are recognized by the blanks around it, and the end of a sentence by the punctuation.

In other cases, elements such as phrases that need to be emphasized (highlighted) or quotations will need to be identified by tags, because these elements are of a different type from the other easily recognized elements in the paragraph unit, such as words and sentences. Tagged elements that occur within the bounds of a paragraph unit are called *text items*. Figure 4 on page 12 shows the structure of a paragraph unit and the types of elements, including text items, that can occur within it.

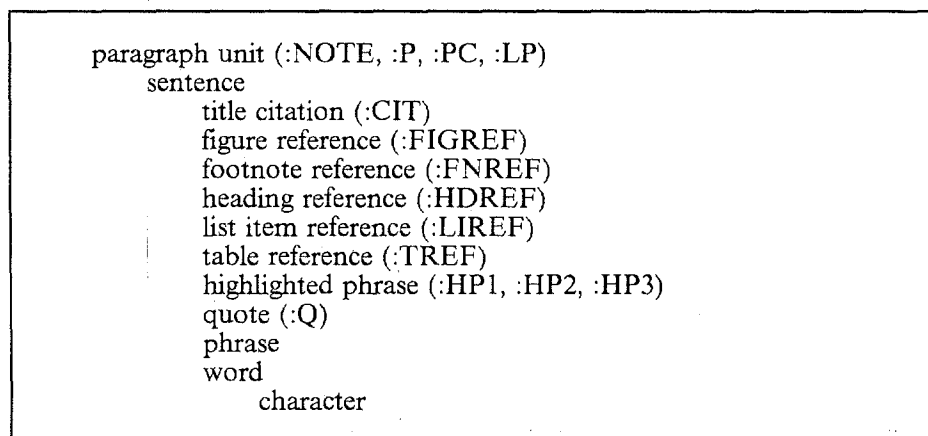


Figure 4. Contents of a Paragraph Unit. Tagged elements that can occur within a paragraph unit are known as text items. The tags used to identify them are shown in parentheses after the element type.

Of course, not every paragraph unit will have all of the elements shown in the figure. In fact, a paragraph unit could contain as little as a single word, or even a single character.

Most often, though, we use tags to identify paragraph units and higher-level elements that are made up of paragraph units. One group of elements for which you will need tags is called *basic document elements* because of the frequency with which they occur. Their tags are the ones you will use most often. They are used for identifying such things as:

- Paragraphs
- Highlighted phrases
- Quotations
- Lists
- Tables.

The basic document elements are listed in Figure 5 on page 13, together with the tag used to identify the element, an indication of the types of element each can contain, and the content of the element.

- address (:ADDRESS)
 - address line (:ALINE): single text line
- definition list (:DL)
 - definition term heading (:DTHD): single text line
 - definition description heading (:DDHD): single text line
 - definition term (:DT): single text line
 - definition description (:DD)
 - implied paragraph
 - basic document elements
 - list part (:LP)
 - implied paragraph
- example (:XMP)
 - basic document elements
 - line elements
- figure (:FIG)
 - figure body
 - basic document elements
 - line elements
 - figure caption (:FIGCAP): single text line
 - figure description (:FIGDESC)
 - implied paragraph
 - basic document elements
- footnote (:FN)
 - implied paragraph
 - basic document elements
- glossary list (:GL)
 - glossary term (:GT): single text line
 - glossary definition (:GD): implied paragraph
 - implied paragraph
 - basic document elements
 - list part (:LP)
 - implied paragraph
- lists (:SL, :OL, :UL)
 - list item (:LI)
 - implied paragraph
 - basic document elements
 - list part (:LP)
 - implied paragraph
 - basic document elements
- long quotation (:LQ)
 - basic document elements
- note (:NOTE): paragraph unit
- paragraph (:P): paragraph unit
- paragraph continuation (:PC): paragraph unit
- table (:TABLE)

Figure 5. Basic Document Elements. The tags that identify each element are in parentheses after the element type. The contents of a paragraph unit are shown in Figure 4. An implied paragraph is one for which no :P tag is entered. The contents of a table are shown in Figure 8 on page 18.

Notes:

1. A single text line must appear entirely on the same line with its tag or entirely on the line immediately following the tag. A single text line cannot contain any tags.
2. A line element appears within examples and figures. The line element appears in the output exactly as it did in the source document, because the text is unformatted within examples and figures. The line element can contain tags.

Some basic document elements are paragraph units. Others are made up of other tagged elements that can themselves be paragraph units, or can contain still other tagged elements. For example, a list element contains list items and list parts. A list item is actually an implied paragraph, but it can also contain most other basic document elements, including lists. Such *nested* structures occur quite frequently. They are easy to mark up with the GML starter set, because you can always use the same tags, regardless of the level of nesting.

Those elements that have an implied paragraph structure are shown in Figure 5 as having the content "paragraph." You need not enter a paragraph tag with these tags, because the existence of the paragraph is understood. Tags with implied paragraph structure can also contain other basic document elements. (There are some restrictions, depending on the element type. For example, footnotes cannot appear within other footnotes, figures, examples or tables. However, footnote references can appear within these elements.)

Chapter 3, "GML Starter Set Tags" on page 35 contains detailed explanations of all element types, the tags that identify them, where they can occur, and what they can contain.

Figure 6 shows the structural elements of a general document

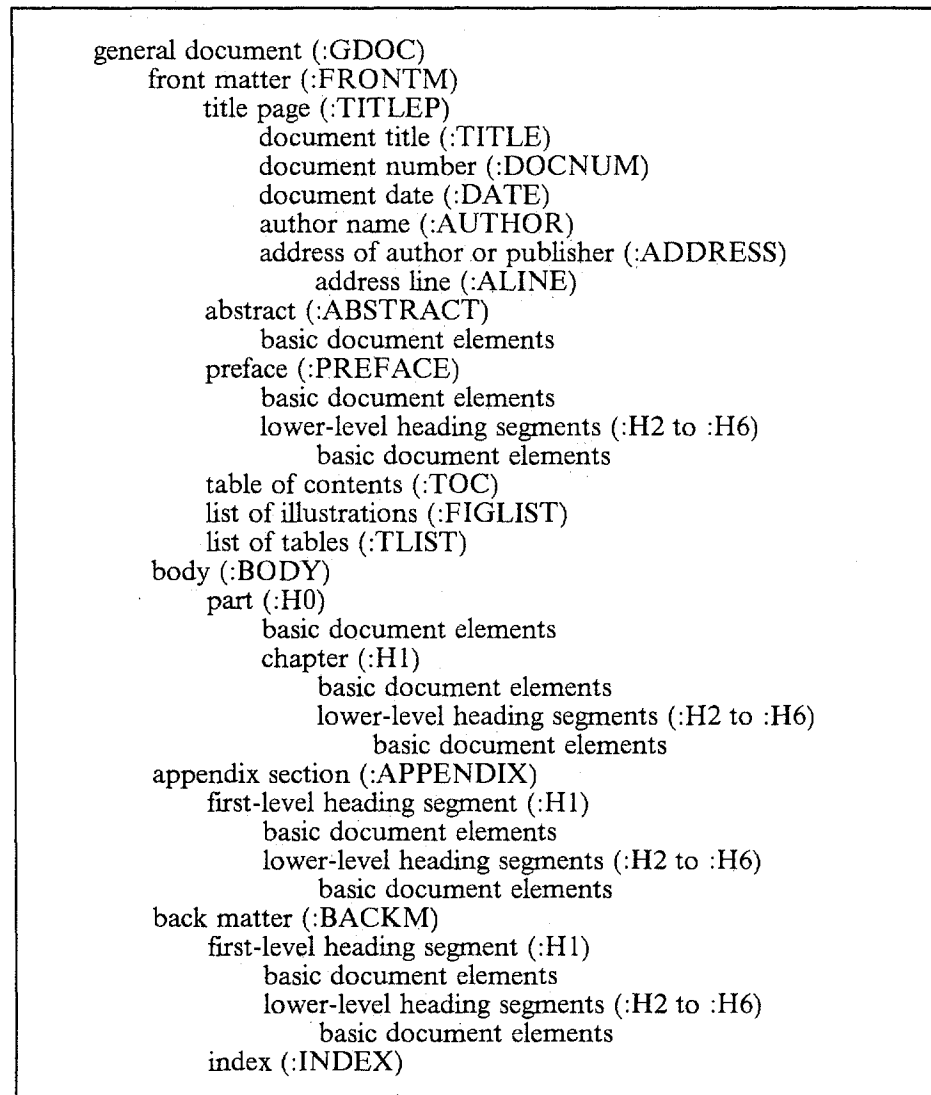


Figure 6. Structure of a General Document. The tags that identify each element are given in parentheses after the element type. Basic document elements are shown in Figure 5 on page 13.

and shows that it consists of four major elements: front matter, body, appendix, and back matter. The structural elements control how the formatting of the page itself is handled; for example:

- :FRONTM (the front matter tag) sets lowercase Roman numerals for page numbering and one-column format for the text, regardless of what is requested on the SCRIPT command.
- :BODY (the body tag) starts Arabic numbering and formats the text as requested on the SCRIPT command.
- :APPENDIX (the appendix tag) causes level-one headings to be lettered serially and keeps the same format as the body.
- :BACKM (the back matter tag) sets the text in two-column format, regardless of what is requested on the SCRIPT command.

The front matter contains specialized elements (such as title page and abstract) that do not occur elsewhere in the document.

The body, appendix, and back matter contain *heading segments*. Heading segments are elements that begin with a heading, followed by basic document elements and other heading segments.

Heading tags are numbered to show the level of segment they begin: :H0 for *zero-level heading segment*, :H1 for *first-level*, and so on to :H6. (A zero-level heading segment is often called a *part*. A first-level segment, when it occurs within the bounds of the body, is known as a *chapter*.) When the document is printed, each level of heading will normally be formatted in a different style to emphasize the structure of the text.

You will find that the more familiar you become with the structure of general documents, the easier it will be to remember the tags and the places where you can enter them in a document.

Some elements are not part of the structure of a document. These elements, listed in Figure 7, can occur anywhere in a document.

index entry (:I1, :I2, :I3): single text line index header (:IH1, :IH2, :IH3): single text line index reference (:IREF) process-specific control (:PSC)
--

Figure 7. Miscellaneous Elements. These elements are not part of the structure of a general document. They can occur anywhere in the document.

Elements in a General Document

This section gives examples of markup and processing for some of the elements of a general document.

Lists

There are several types of list elements. They are simple, ordered, and unordered, and are identified by the :SL, :OL, and :UL tags, respectively. Each element within a list is identified by a :LI tag. Lists can be nested—a list can appear within a list item of another list.

You can nest as many lists as you need. Because each nested list is indented from the left edge of the previously nested list, the only restriction is the width of your column.

Each simple, ordered, or unordered list must be ended by a :ESL, :EOL, or :EUL tag, respectively.

The items of a simple list (:SL) look like this:

- A simple list item
- Another simple list item
- Yet another simple list item.

The items of an ordered list (:OL) are alternately numbered and lettered, based on the level of nesting:

1. Level one
 - a. Nesting level two
 - 1) Nesting level three
 - 2) Nesting level three
 - b. Nesting level two
2. Level one

The items of an unordered list (:UL) are preceded by an identifier such as a bullet or dash. The identifier used depends on the device the formatted output is printed on.

- Level one
 - Nesting level two
 - ▲ Nesting level three
 - ▲ Nesting level three
 - Nesting level two
- Level one

You can intermix and nest different types of lists in any combination.

Following is an example that nests a combination of ordered, unordered, and simple lists.

1. Level one is an ordered list item.
2. This is another ordered list item at level one.
 - The second level of listing has been changed to an unordered list.
 - Even though it's at the second level of listing, because it's the first use of an unordered list, it still uses the first-level unordered list item identifier—the bullet.
 - A simple list follows. No matter where you see a simple list, it always looks the same.

This is the third level of listing.

Keep list items short in a simple list.

3. Both the simple list and unordered list have been ended, and this is the last item of the ordered list.

Definition Lists

Definition lists are identified by the :DL and :EDL tags; the terms being defined are identified by :DT tags, and the definitions are identified by :DD tags. You can also identify headings for definition lists. The heading for the definition terms column is identified by the :DTHD tag. The heading for the definition descriptions column is identified by the :DDHD tag. A definition list with headings looks like this:

<i>TERMS</i>	<i>DESCRIPTIONS</i>
---------------------	----------------------------

Term One	Definition description for first term.
----------	--

Term Two	Definition description for second term.
----------	---

Examples

Example tags are identified by the :XMP and :EXMP tags. They differ from figures in that examples are neither captioned nor referred to from other parts of the document. The font used for an example is a uniformly spaced font. An example looks like this:

This is an example
of an example.

Glossary Lists

Glossary lists are identified by the :GL and :EGL tags. The glossary terms are identified by the :GT tag and the glossary definitions are identified by the :GD tag. These glossary tags were used in preparing the "Glossary" on page 161.

Figures

Figures, such as a diagram or other illustration, are identified by the :FIG and :EFIG tags. Figures might also contain a figure caption and a figure description, identified by the :FIGCAP and :FIGDESC tags, respectively. Figure 7 on page 15 is an example of a figure that has both a figure caption and a figure description.

Tables

Tables are identified by the :TABLE and :ETABLE tags and must be preceded by an :RDEF tag. The :RDEF tag defines the characteristics of a row in a table. Tables might also contain a table caption and a table description, identified by the :TCAP and :TDESC tags. Figure 8 contains a complete list of the contents of a table.

```
table (:TABLE)
  table header (:THD)
    cell (:C)
      implied paragraph
      basic document elements
  table footer (:TFT)
    cell (:C)
      implied paragraph
      basic document elements
  row (:ROW)
    cell (:C)
      implied paragraph
      basic document elements
  table caption (:TCAP): single text line
  table description (:TDESC)
    implied paragraph
    basic document elements
  table note (:TNOTE)
    implied paragraph
    basic document elements
```

Figure 8. Contents of a Table. The :RDEF tag must precede a :TABLE tag.

Footnotes

Footnotes are identified by the :FN and :EFN tags.³

³ This is an example of a footnote.

Highlighted Phrases

Certain elements of a document can be emphasized, or *highlighted*, to contrast with text that is not emphasized when printed. Three levels of highlighting are provided with the starter set. They are: *highlight 1*, **highlight 2**, and *highlight 3*. Text that is not emphasized is highlight level 0. The appearance of the highlighting depends on the device used for printing. Highlighting can be done with capitalization, underscoring, overstriking, or a change of fonts. The tags used to identify the highlighted phrase, however, remain the same.

Highlighted phrases in a document are identified by the :HP1, :HP2, and :HP3 tags. The :HP0 tag can be used to de-emphasize a part of a highlighted phrase. You need to mark the end of a highlighted phrase with a :EHP1, :EHP2, :EHP3, or :EHP0 tag.

Cross-References

Footnotes, tables, figures, index entries, list items, and heading elements each have an attribute called a *unique identifier*. The value of the unique identifier, specified with the ID attribute, is a string of up to seven numbers and upper-case or lowercase letters that can be used to refer to the element. IDs are *case sensitive*. The ID "abc" is different from the ID "ABC." Therefore, when referring to an element, it is necessary to know the exact ID. No two elements of the same type can have the same identifier.

For example, a footnote and a figure, each of which has a unique identifier, can be entered as:

```
:fn id=xmpl.  
:  
:efn.  
  
:fig id=fig1.  
:  
:efig.
```

The tags :FIGREF, :TREF, :FNREF, :IREF, :LIREF, and :HDREF are used to refer to uniquely identified figures, tables, footnotes, index entries, list items, and headings, respectively. Each of these tags inserts into your document a reference to the element you identify with the REFID attribute. (Ensure your REFID is the ID you are referring to.) The following fragment of a source document contains references to the footnote and a figure previously identified:

```
... fully,:fnref refid=xmpl. as  
illustrated in :figref refid=fig1..
```

The GML starter set formats this fragment as:

... fully,³ as illustrated in Figure 4 on page 12.

Index

A subject index can be produced from index terms placed throughout the document. Three levels of index terms can be identified with the :I1, :I2, and :I3 tags. Index headers (terms without page references) can be identified with the :IH1, :IH2, and :IH3 tags. An index entry can have more than one page reference after it without being typed again by using the :IREF tag.

When the :INDEX tag is encountered, an index is built from the terms specified with these tags. Page references are automatically provided for the index terms.

Note: In order to get an index, you must specify the INDEX command option.

Refer to the *Document Composition Facility: Generalized Markup Language Starter Set User's Guide* for a list of common index sorting conventions. The sort sequence for each language includes the special sorting requirements of that language. Refer to the *Document Composition Facility: SCRIPT/VS Text Programmer's Guide* for additional information on the sort sequence for each supported language.

Entering GML Markup

A document is marked up by identifying its elements. It is almost always necessary to mark the start of an element explicitly. The rules for GML markup are:

- Start all lines at the left margin of the source document. There should be no leading blanks. Leading blanks cause the SCRIPT/VS .LB [Leading Blank] control word to be executed. Trailing blanks at the end of the input line are ignored.
- An element of a document is marked up with a *tag* preceded by a special character called the GML delimiter (:). Some examples of tags are :FIG, :GDOC, :TITLE, and :H1.

Sometimes it is also necessary to mark the end of an element explicitly. This is done with the same tag used at the start, but preceded by the GML delimiter and an *e* (for end). Some examples are :EFIG and :EGDOC.

Begin a new input line for each tag except text items, such as :HP0 - :HP3 for highlighted phrases or :Q for quotations.

Note: In this document, tags are usually shown in capital letters. However, you can enter tags in either uppercase or lowercase, whichever is more convenient for you.

- Some tags have *attributes* that expand that tag's function and are entered after the tag name:

:tag attribute=value.

At least one blank must separate the attribute name from the tag name. If the value contains blanks or special characters, it must be enclosed in single quotation marks ('), and if the value itself contains a single quotation mark, you must double it:

:tag att='Mrs. O''Grady's cat'.

- Some tags have *value attributes*, which consist of a single word, after the tag name:

```
:tag value.
```

- When a tag has several attributes, you can enter them all on a single line with the tag, or you can place them on several successive lines:

```
:tag att1=value1 att2=value2
att3=value3 att4=value4
att5=value5.
```

You can enter as many attributes and values on a line as you want, and on as many lines as you need; however, each attribute and its value must be kept together on the same input line.

- The last attribute (or the tag, if there are no attributes) should be followed by a *markup/content separator* (.):

```
:tag attribute=value.
```

```
:tag.
```

The markup/content separator is not always necessary. For example, when a tag is immediately followed by another tag, there is no need to indicate the end of the markup. However, it is never wrong to include a markup/content separator. If in doubt, put one in.

- Some tags perform immediate processing on the text that follows the markup/content separator. This text must be entered all on the same line with the markup/content separator or all on the line after the end of markup, and it cannot contain any other tags. Some examples are the :H0 – :H6 tags and the :DT tag.

```
:tag.following text
```

```
:tag att1=value1 att2=value2
att3=value3.following text
```

- Some attributes call for a unique identifier as their value. Unique identifiers are up to 7 uppercase or lowercase letters and numbers. These are the ID and REFID attributes.
- Some attributes call for an amount of vertical or horizontal space as their value. Any of the notations shown in Table 1 on page 22 can be used to specify an amount of space. Be advised, however, that device units vary greatly among printers. Using specific device units can make your document device dependent.
- When you use a period to enter *decimal* values on attributes, enclose the value within a set of single quotation marks. If single quotation marks are not used, the starter set will interpret the period as a *markup content separator*, not as a decimal point, so the attribute value will not be processed correctly.
- Because of the way DCF processes residual text, the residual text for a GML tag must not exceed 250 characters. If the residual text exceeds 250 characters, the line is truncated and a message is issued. For more information on residual text, refer to Chapter 1 in the *Document Composition*

Space Unit	Specified As	Examples
Centimeter	aCM	4.25cm 2,54cm 15cm
Character (Horizontal)	a	5 12.5 1,33
Cicero	nCp	c12 (12 didot points) 2c3 (2 Ciceros and 3 didot points) c1.5 (1.5 didot points)
Device Unit (Horizontal)	nDH	10dh 600dh
Device Unit (Vertical)	nDV	10dv 600dv
Em space (Horizontal)	aMH or aM	6mh 6m .33mh .33m
Em space (Vertical)	aMV	1mv .5mv
Inch	aI	3.5i 6,5i .75i
Line (Vertical)	a	2 3.5 1,75
Millimeter	aMM	12.7mm 25,4mm 100mm
Pica	nPp	p6 (6 points) 3p2 (3 picas and 2 points) p1.5 (1.5 points)
<p><i>Where:</i></p> <p>a is a number of centimeters, characters, ems, inches, lines, or millimeters. The number can be fractional, with up to two decimal positions. Either a period (.) or comma (,) can be used to separate the integral and fractional portions of the number.</p> <p>n is a number of whole ciceros, picas, or device units.</p> <p>p is a number of points. (There are 12 didot points in a cicero, 12 points in a pica, and 72 pica points in an inch.)</p> <p>Warning: All space units, except device units, are subject to rounding and truncation. Refer to <i>Document Composition Facility: SCRIPT/VS Text Programmer's Guide</i> for more information.</p>		

Table 1. Space-Units Notation

Entering Text

The following practices are recommended when entering text:

- Start all lines at the left margin of the source document. There should be no leading blanks. Leading blanks cause the SCRIPT/VS .LB [Leading Blank] control word to be processed. Trailing blanks at the end of an input line are ignored.
- Don't use blanks to try to line up text in a column; use tabs to line up text in a column. Refer to the *Document Composition Facility: SCRIPT/VS Language Reference* for information on using tabs.

- Don't use the tab key. Indentation is usually used to set off an example or a list of items. In these cases, you should mark the text with a tag that describes it, such as :XMP or :LI.
- Begin each new sentence on a new line. When you end a sentence, do not enter any more text on that line. SCRIPT/VS will automatically add two blanks between sentences. Don't end a line with a hyphen.
- Don't begin a line with a period unless you are entering a SCRIPT/VS control word. A control word is identified by a period in the first position of an input line.

You should use only the control words that are listed in this chapter under the section, "Control Words and Macros" on page 26. Because the APFs that make the GML tags work are built using control words, you could cause a conflict between control words that would produce unexpected results.

- Don't enter any blank lines. (This applies to *null* lines and *index returns* as well, for those keyboards that ordinarily permit them.)
- Keep your input lines short, for ease of revision.
- Don't underscore or overstrike any characters. Use the tag for the appropriate level of highlighting instead.
- For convenience in handling, divide a large document into several source documents and store each part in a separate computer file. Ensure that each file starts at the beginning of a paragraph unit or higher-level element and ends at the end of one.

Note: Some of the foregoing practices might not apply when entering text and control words as part of a :PSC element, an example, a figure, or a table.

Text Associated with a Tag: Once SCRIPT/VS has identified the markup for a tag, it identifies the text associated with the tag. This text is called the residual text. Not all of the tags in the starter set use residual text, but SCRIPT/VS checks for it anyway. The residual text starts right after the markup content separator (.) for the tag. The end of the residual text is defined as the end of the input line, or the occurrence of another GML tag on that input line. If there is nothing on the input line after the tag markup, the residual text starts on the next input line. It is ended by either the end of that input line or the occurrence of another GML tag on that input line.

SCRIPT/VS Symbols

A symbol is a string of characters that begins with an ampersand (&) and ends with a period (.) or a blank. Unlike GML tags, however, SCRIPT/VS makes a distinction between uppercase and lowercase characters in the symbol name. When you refer to a symbol in your source document, it must be the same case as was used when the symbol was identified. For example, &case and &CASE are two separate symbols to SCRIPT/VS. When you include a symbol in a source document, SCRIPT/VS replaces it with other previously identified characters in the output document. Figure 9 on page 25 shows the markup for this section of the book and demonstrates the use of symbols.

Your installation may have defined symbols for such purposes as:

- Entering characters that are not available on your keyboard
- Abbreviating frequently used lengthy phrases

- Avoiding confusion with characters that have special meaning to SCRIPT/VS.

A symbol can be entered anywhere in a document where the characters the symbol represents could have been entered. One exception to this rule is, GML tag names cannot contain symbols. You can, however, use a symbol within the value of an attribute. The starter set provides the following symbols:

& This symbol is replaced with an ampersand (&). Use it instead of the ampersand whenever it is immediately followed by a letter, a number, or the characters @, \$, or #. This prevents the text from being interpreted as a symbol.

&date. This symbol is replaced with the current date, or, if one was given on the :DATE tag, the document date. You can use it whenever you want to include the document date in your formatted output. For example, this document was formatted on February 13, 1991.

The preceding sentence was entered as:

For example, this document
was formatted on &date..

The format of the date is determined by the value specified on the LANGUAGE attribute of the GDOC tag, or the default language at your installation if a language is not specified.

&gml. This symbol is replaced with the GML delimiter (:). Use this symbol in examples of GML markup instead of the real GML delimiter character. This prevents the markup examples from being interpreted as real markup.

&rbl. This symbol is replaced with a *required blank*. When you want adjacent words to be treated as a single word to keep them on the same output line, use &rbl. to separate them, instead of the ordinary blank. You can repeat the &rbl. to create any number of blank spaces. Each &rbl. leaves one blank space.

&semi. This symbol is replaced with a semicolon (;). SCRIPT/VS treats the (;) as a control word separator. If your document contains examples of SCRIPT/VS control words separated with semicolons, use this symbol where you want the semicolons to appear.

&time. This symbol generates the current time. You can use it whenever you want to include the time at which your document is formatted. For example, this document was formatted at 9:35 a.m.

The preceding sentence was entered as:

For example, this document
was formatted at &time.

The format of the time is determined by the value specified on the LANGUAGE attribute of the GDOC tag, or the default language at your installation if a language is not specified.

If you enter a symbol incorrectly, it will be treated as text, unless the error results in the name of another symbol.

Note: Your document administrator should be able to advise you of other symbols you might find useful in your documents.

:h2 id=symbol. SCRIPT/VS Symbols

:p.A symbol is a string of characters that begins with an ampersand (&) and ends with a period (.) or a blank. Unlike GML tags, however, SCRIPT/VS makes a distinction between uppercase and lowercase characters in the symbol name. When you refer to a symbol in your source document, it must be the same case as was used when the symbol was identified.

For example,

&case and &CASE are two separate symbols to SCRIPT/VS.

When you include a symbol in a source document, SCRIPT/VS replaces it with other previously identified characters in the output document.

:Figref refid=se. shows the markup for this section of the book and demonstrates the use of symbols.

:p.Your installation may have defined symbols for such purposes as:

:ul.

:li.Entering characters that are not available on your keyboard

:li.Abbreviating frequently used lengthy phrases

:li.Avoiding confusion with characters that have special meaning to SCRIPT/VS.

:eul.

:p.A symbol can be entered anywhere in a document where the characters the symbol represents could have been entered.

One exception to this rule is, GML tag names cannot contain symbols.

You can, however, use a symbol within the value of an attribute.

The starter set provides the following symbols:

:dl tsize=8.

:dt.&.

:dd.This symbol is replaced with an ampersand (&).

Use it instead of the ampersand whenever it is immediately followed by a letter, a number, or the characters @, \$, or #.

This prevents the text from being interpreted as a symbol.

:dt.&.date.

:dd.This symbol is replaced with the current date,

or, if one was given on the :DATE tag, the document date.

You can use it whenever you want to include the document date in your formatted output.

:

:dt.&.time.

:dd.This symbol generates the current time.

You can use it whenever you want to include the time at which your document is formatted.

For example, this document was formatted at &time.

:p.The preceding sentence was entered as:

:xmp.

For example, this document was formatted at &time.

:exmp.

:edl.

:p.If you

enter a symbol incorrectly, it will be treated as

text, unless the error results in the name of another symbol.

:note.Your document administrator should be able to advise you of other symbols you might find useful in your documents.

Figure 9. Markup Example. The source for the start of the section "SCRIPT/VS Symbols" on page 23.

Control Words and Macros

Some situations occur in which it may be necessary to supplement GML markup with SCRIPT/VS control words or macros. Control words are the command language of SCRIPT/VS. Macros are processing programs written by a text programmer in the SCRIPT/VS language. The rules and conventions for entering control words also apply to macros. Use control words and macros to:

- Manage source documents
- Obtain specific formatting results not provided by tags
- Modify or tune processing results
- Supplement the general processing specified in the profile
- Reserve space in a document for later insertion of variable text.

Control words and macros must be entered at the left margin of your source document, preceded by a period (.). Some control words and macros require information, called *parameters*. Parameters are entered on the same line as the control word or macro, separated from each other and from the control word or macro name by blanks.

Note: You must enter control words in accordance with the procedures in this section. Although not doing so might produce output, nonstandard practices could make it more difficult to maintain the document and to use it for other applications. You can follow these procedures and still obtain all of the functions that direct entry of control words can provide.

Source Document Management

For convenience in managing your documents, you may want to:

- Break tables to save storage for processing
- Store parts of your document as separate files and have SCRIPT/VS combine them for processing
- Define symbols for lengthy or frequently used words or phrases
- Write comments to readers of your source document
- Identify revised portions of the text
- Control the way index terms are sorted
- Add words to the spelling dictionary
- Indicate a place in your source document where you may insert text at a later time
- Include a picture or image.

The following control words will help you handle these situations.

Table Storage Relief (.TABREAK)

Use the TABREAK macro to save storage while formatting tables. The TABREAK macro ends the current row in a table and causes the rows that have been formatted to be placed. TABREAK also causes a column or page eject, and returns the storage that has been used for formatting that portion of the table. It then continues processing with the current header and footer. It does not end the table.

Imbedding Files (.IM)

The .IM [Imbed] control word causes the file whose file-identifier is given as the parameter to be processed at this point, as though the current file and the imbedded file were a single file. Many different files can be imbedded to form a single document.

For example:

```
.im intro
```

causes the file named INTRO to be processed at the point where the .IM control word is entered. Processing of the file containing the .IM control word continues after the imbedded file has been completely processed.

Note: In TSO, you must specify a .DD control word, because the fully qualified data set name option is not available.

Defining Symbols (.SE)

You can define a SCRIPT/VS symbol for frequently used words or lengthy phrases. Wherever you enter the symbol, SCRIPT/VS replaces it with its value. Symbol names are composed of up to ten numbers and uppercase or lowercase letters. Symbols are case sensitive. The same name can be used for different symbols, that is, you can specify a word in lowercase letters as having one value and the same word in uppercase letters as having a different value. Symbols are defined with the .SE [Set Symbol] control word, for example:

```
.se top = 'top of the page'  
.se TOP = 'Two of Perry's'  
.se toP = 'tables on Prices'
```

If the symbol value contains blanks or special characters, it must be enclosed in single quotation marks; if the symbol value contains single quotation marks, they must be doubled.

You can use a symbol anywhere in your document where you would otherwise have entered its value. When you use the symbol, you must precede it with an ampersand (&) to indicate to SCRIPT/VS that what follows is a symbol name. You should also follow the symbol name with a period (.). The period following a symbol name, like the markup/content separator after a tag, is not always required. However, it is never wrong to include it. If the symbol represents a word to be followed by punctuation, you must put in the period before the punctuation, or the symbol will not be recognized. If you end a sentence with a symbol, you must put in two periods: one to end the symbol and one to end the sentence. So, if you enter a sentence using the symbols just defined above, you would enter it like this.

```
&TOP. &toP. will not fit on  
the &top..
```

When printed, it looks like this:

Two of Perry's tables on Prices will not fit on the top of the page.

Writing Comments (.*)

To write a comment to readers of the source document (such as yourself), enter:

```
. * remember to finish this section  
. * (you may enter as many comments  
. * as you like, but each line  
. * must begin with .*)
```

Comment lines are ignored by SCRIPT/VS.

Note: Comment lines can also be marked with the control word `.CM`, but the use of `.*` is recommended. SCRIPT/VS examines all control word lines for the control word separator (`;`) and treats the text following the control word separator as a new input line. Lines beginning with `.*` are not so examined.

Indicating Revisions (.RC)

You can call attention to a revised portion of a document by flagging the changed lines with revision codes either to the right or left of your text. The `.RC` [Revision Code] control word is used for this purpose. You can use the `.RC` control word to define up to nine characters that can be used as markers to indicate changes in your document. You can also specify the name of a font to be used for printing the revision code character. Revisions in documents are indicated by turning on a previously defined revision code and then turning it off following the last line of new input. A typical revision code is the vertical bar (`|`). It must be defined first and then turned on and off with the `.RC` control word.

Note: Do not adjust or change the alignment of revision codes with the `.RC ADJUST` or `.RC ALIGN` control words. (Consult your document administrator for the procedures for using revision codes at your installation.)

Controlling Index Sorting (.DC)

By using the `IXI` parameter of the `.DC` [Define Character] control word, you can specify selected characters, such as blanks, to be ignored when sorting.

```
.dc ixi 40 "
```

This causes blanks and double quotation marks to be ignored during sorting. To specify a blank for this control word, you must use the hexadecimal code (40) instead of entering a space.

You can also specify selected characters that you want to be treated the same as blanks by using the `IXB` parameter of the `.DC` control word. For example,

```
.dc ixb *
```

causes asterisks (`*`) to be treated the same way as blanks for sorting purposes.

Note: For more information on index sort sequence, see your text programmer or refer to the *Document Composition Facility: SCRIPT/VS Text Programmer's Guide*.

Adding Words to the Dictionary (.DU)

You can create a dictionary to supplement the main spelling dictionary for both SPELLCHK and hyphenation.

Use the .DU ADD control word to build an addenda dictionary. For example:

```
.DU ADD elon-ga-tion has-ten  
.DU ADD si-mul-ta-ne-ous GML
```

As many words as can fit can be given on a single control word line. Separate each word by a blank. Use uppercase letters only when they are required, such as in a name. Specify the appropriate hyphenation points for each word as you list it. Each word is divided into four three-character groups following the first vowel, with only one hyphenation point recorded for each of the four groups. Refer to the *Document Composition Facility: SCRIPT/VS User's Guide* for additional details on hyphenation.

Note: For a normally hyphenated expression such as lighter-than-air, enter two hyphens with the .HW control word to ensure that SCRIPT/VS will print the required hyphens.

If you are building this dictionary to be used with more than one document, create a separate file to contain the .DU control words. You can then imbed this file at the beginning of each input file that requires it.

Reserving Space for Variable Text (.VT)

The .VT [Variable Text] control word is used to indicate a place in your source document where you can insert text at a later time. When the document is ready to include the variable text, the appropriate postprocessor for the environment and the data must be run. The postprocessors work only on documents formatted for page-mode devices. For more information, refer to the *Document Composition Facility: SCRIPT/VS Language Reference*.

Including a Page Overlay (.OI)

When formatting for AFP devices (except the 3800 Printing Subsystem Model 3 and 6), you can include an overlay in your document. You must identify the file containing the page overlay in your source document with the .OI [Overlay Include] control word.

SCRIPT/VS doesn't access the page overlay, but it must be available at print time. In MVS, the overlay name must be a member of the partitioned data set defined as the overlay library. In CMS, the default filetype for overlays is OVLY38PP. In VSE, page overlays are not supported.

You can also include overlays in your document by specifying .CG [Copy Group] to change copy groups in your FORMDEF. Refer to the *Document Composition Facility: SCRIPT/VS Language Reference* for information on .CG [Copy Group], and the *Document Composition Facility: SCRIPT/VS User's Guide* for more information on using copy groups to include overlays.

Overlays may be included at the current position with space reserved or at a specified position without reserving space.

Reserving Space for a Page Overlay

If you want to reserve space for an overlay, you can specify the DEPTH and WIDTH parameters on the .OI [Overlay Include] control word. These parameters reserve space for an overlay to be included at the current print position. As an example, you might specify:

```
.oi olrose width 5i depth 3i
```

which reserves 5 inches of horizontal space and 3 inches of vertical space for the proposed page overlay. ("olrose" is the overlay name.) It is important that you reserve enough space for the actual size of the overlay, or it will overlap the text that follows the reserved space.

If you are using a device type that does not support overlays, the overlay name will be ignored and the space specified on the DEPTH and WIDTH parameters will be reserved. This allows you to reserve space for cut-and-paste artwork.

Overlays included with .OI [Overlay Include] will be included only on the current page. You can use the .OI control word to produce a multi-page overlay by including it within a running header or footer.

Refer to the *Document Composition Facility: SCRIPT/VS Language Reference* for more information on the .OI [Overlay Include] control word.

Positioning a Page Overlay (.OI)

You can also include a page overlay in your document by using the .OI control word with the HPOS and VPOS parameters. These parameters position an overlay at a specific location on a page, but do not reserve space. The top left center of the overlay is placed at the specified horizontal and vertical position regardless of the position of any other text, graphics, or images. As an example, you might specify:

```
.oi olrose hpos 1i vpos 1i
```

which positions the overlay on the page 1 inch horizontally and 1 inch vertically from the logical page origin. ("olrose" is the overlay name.)

Overlays included with .OI [Overlay Include] will be included only on the current page. You can use the .OI control word to produce a multi-page overlay by including it within a running header or footer.

Refer to the *Document Composition Facility: SCRIPT/VS Language Reference* for more information on the .OI [Overlay Include] control word.

Including a Page Segment (.SI)

When formatting for IBM page devices, you can include a page segment in your document that can be a picture, logo, or some other image. You must identify the file containing the page segment in your source document with the .SI [Segment Include] control word. To include images when formatting for PostScript devices, use the .PO [PostScript] control word (see "Including PostScript Images (.PO)" on page 32 for more information).

SCRIPT/VS doesn't format the page segment. It saves a place in the formatted document, and the page segment is included when the document is printed.

Use the :FIG tag to do this when you want to add a caption to the page segment included. For example,

```
:fig.  
.si pic  
:figcap.Three-Toed Sloth  
:efig.
```

The image contained in the file named "pic" will be included in the document when it is printed on a page printer, provided the page segment is available in the appropriate segment library.

Page segments are not interchangeable between the 4250 Printer and other page printers.

If the page segment you request is not in the default page segment library, you must specify the SEGLIB option on the SCRIPT command.

Reserving Space for a Page Segment

Sometimes, when you are working on a draft of a document, for example, you may want to reserve space for a page segment that is incomplete or that does not yet exist. You can reserve this space by specifying a width, a depth, or both. As an example, if your document were one-column and you expected the page segment to take up a large part of the output page, you might specify:

```
.si clash width 6i depth 7.5i
```

which reserves 6 inches of horizontal space and 7.5 inches of vertical space for the proposed page segment. Later, when the actual page segment is included in your document, any width and depth values you have specified will be replaced by the actual size of the page segment as it has been specified in the segment library unless you specify ABSOLUTE.

The ABSOLUTE parameter allows you to override the actual dimensions of the segment⁴ with the WIDTH and DEPTH specified on the .SI [Segment Include] control word.

See Chapter 4, "Processing GML Documents with SCRIPT/VS" on page 117 for descriptions of the SEGLIB and NOSEGLIB options of the SCRIPT command.

Note: Whenever you include page segments in your document, if the document will be printed on:

- An MVS system, the name specified with the .SI [Segment Include] control word must be the name of a member in a segment library
- A CMS system, the name specified with the .SI [Segment Include] control word must be a CMS file name
- An ATMS-III system, the name specified with the .SI [Segment Include] control word must be the name of a member in a segment library

⁴ For AFP and some 4250 segments, the size is based on the size of the first image object only. If the segment contains multiple image blocks or composed text, it may be larger than SCRIPT/VS expects.

- A TSO system, the name specified with the .SI [Segment Include] control word must be the name of a member in a segment library
- A VSE system, segment libraries are not searched by DCF, but can be used at print time.

Refer to the *Document Composition Facility: SCRIPT/VS User's Guide* for more information concerning page segments.

Including PostScript Images (.PO)

If you plan to print your document on a PostScript device, you can include PostScript image files in the document. PostScript image files are files containing graphics or text that are created using PostScript page-description language. PostScript image files are included in a DCF document by using the .PO [PostScript] control word.

The .PO control word identifies the place in the document where you want the PostScript image to be included and allows you to specify:

- Whether or not the image is placed within a line of text
- The width and depth of the image
- The dimensions by which you want the image to be scaled (made larger or smaller).

To learn how to use the .PO control word, refer to the *Document Composition Facility: SCRIPT/VS User's Guide*.

If you want an included PostScript image to have a caption, you can specify that the image be included within :FIG start and end tags like this:

```
:fig.  
.po imagel  
:figcap.Wapiti  
:efig.
```

Note: You must specify a PostScript device type on the SCRIPT command to format your document so that PostScript images are included, and you must print the document on a PostScript device (the IBM 4219 Personal Pageprinter, for example).

Graphic Formatting

Your document might require formatting that is not conveniently available with the APFs provided by your installation. For example, you might want to use character graphics to create an illustration, such as a flowchart or a complex table. Illustrations can be done with specific markup. However, different versions of the marked-up illustration might be required for different processing that can be performed on the document. For example, producing a single-column draft on a 3270 terminal would be a different process from producing a two-column final copy on an IBM 3800 Printing Subsystem.

To make this flexibility possible, enter such formatting control words only within a special element called a *process-specific controls* element (:PSC). The PROC attribute of the :PSC tag lets you specify the processes to which it applies.

When a :PSC element is used in this way, it is called a *graphic*. As the following example illustrates, you can have more than one version of a graphic.

SCRIPT/VS automatically includes the version for the process being run, and ignores the others.

```
:fig.  
:psc proc='1403'.  
  (formatting control words, data)  
:epsc.  
:psc proc='3800'.  
  (other control words, data)  
:epsc.  
:psc proc='4250'.  
.si diag  
:epsc.  
:figcap.Typical Diagram  
:efig.
```

To mark up a graphic, you must know how to use SCRIPT/VS. For further details, including the list of recognized processes, see the description of “:PSC—Process-Specific Control” on page 85.

Modifying Processing Results

You may occasionally be dissatisfied with the results produced by your formatter, whether it be SCRIPT/VS or some other formatter for which SCRIPT/VS is being used as a preprocessor. For example, the formatter can break a column or page at an unsatisfactory place, or it can leave unwanted white space on the page. You would not want to make a permanent change in the document markup to solve such problems because they might disappear when the text of the document is revised. Similarly, they might occur only on one output device and not on others.

The situation is similar to those that require graphic :PSC elements. In both cases, you must tell the system which processes apply. However, there is one significant difference. The graphic element is a permanent part of the document, whereas the modification is only temporary.

For this reason you should enter a temporary process name:

```
:psc proc=patch.  
:epsc.
```

Such :PSC elements will be ignored when you process your document unless you specifically indicate that they are to be processed. “SCRIPT Command Options” on page 117 describes how to tell SCRIPT/VS to process specific :PSC elements with the SYSVAR P option.

Note: Although specific :PSC elements are ignored if you do not specifically enable them, you really should remove them from the source file when they become obsolete, because you may someday revise the file or change your formatting conventions.

Chapter 3. GML Starter Set Tags

The following pages describe the elements of a general document, the tags that identify them, and the processing that is performed by the GML starter set provided with SCRIPT/VS. Each tag description includes a syntax description similar to this:

:TAG	ATTRIBUTE = value $\left[\begin{array}{l} \text{ATTRIBUTE} = \frac{\text{value1}}{\text{value2}} \\ \text{value3} \end{array} \right]$ [value]
------	---

The leftmost column shows the tag name preceded by the GML delimiter (:). When capital letters are used in a syntax diagram, enter the characters shown, but you can use either uppercase or lowercase letters.

Attributes are listed to the right of the tag name. Brackets indicate that the attribute is optional; you need not enter it at all if the attribute is irrelevant to the element you are describing or if the attribute's default value, which is shown underscored, is appropriate.

Attribute values that are shown in lowercase letters describe variable information that you should supply when you enter the attribute. Such text can include uppercase and lowercase characters and must be enclosed in single quotation marks (') if it includes blanks or special characters.

When an attribute can have one of several specific values, those values are shown in a stack. When you use such an attribute, you should pick one of the values shown. If a value is underscored, it is the default and will be used when you do not enter the attribute.

Some attributes consist of only a single word. These *value attributes* can be entered as shown, without any preceding attribute name.

Some tags perform immediate processing on the line of text that follows the markup/content separator (MCS). When the line of text that follows an MCS has particular meaning to that tag, a third column is included in the syntax diagram to describe the meaning of that text:

:TAG		.following text
------	--	-----------------

The following text must be entered on a single line. It must be separated from the tag and its attributes by a markup/content separator and must not contain any other tags. The markup/content separator is used to separate the tag and its attribute from the following text.

Some tags indicate the beginning of a particular document element and have a matching tag that indicates the end of the element. The syntax diagrams for such tag pairs have a second row to describe the end-tag:

:TAG	[ATTRIBUTE = value]
:ETAG	

Below the syntax diagrams are descriptions of the element and each of its attributes, and discussions under the following headings:

Usage: This section discusses where in a document's structure the element can occur, what elements it can contain, and how the element is terminated, together with any special entry rules.

Some elements, such as the author element, can only occur in a specific place in the document. These elements are frequently optional (they need not occur in every document) and repeatable (they can occur in succession). When this is the case, it is mentioned in this section.

Example: This section shows a markup example for each of the tags.

Processing: Markup is the same, regardless of the formatting parameters you specify and the device for which you format a document. Processing, though, depends upon the profile, APFs, and output device, and can vary depending upon options of the SCRIPT command. This section describes the processing of the element, which is performed by the GML starter set.

Formatting Results: The results of the formatting for the example given with each of the tags is shown in this section.

:ABSTRACT—Document Abstract

The :ABSTRACT tag identifies a summary of the document.

:ABSTRACT

Usage: An abstract appears within the front matter of the document, after the title page. It can contain any of the basic document elements listed in Figure 5 on page 13, as well as level-two through level-six heading segments.

An abstract is implicitly ended by a preface, table of contents, figure list, or body element.

Example: The front matter of a document might contain the following markup:

```
:gdoc.  
:frontm.  
:titlep.  
  
:  
:etitlep.  
:abstract.  
:p.This manual describes a  
method by which you can  
obtain the benefits of a powerful  
text-processing system without  
becoming an expert in composition.  
:preface.  
  
:
```

Processing: The :ABSTRACT tag begins a new page. If duplexing is requested with the SYSVAR D option, the new page will be odd-numbered. A level-one heading with the text "ABSTRACT" is placed on the new page.

Note: All the material that appears in the front matter, including the abstract, is in single-column format. The front-matter headings do not appear in the table of contents nor are they numbered. Automatic head-level numbering is suppressed for headings in the front matter, even if requested with the SYSVAR H option. (See "SYSVAR Option" on page 133.)

Formatting Results: The abstract will appear at the front of the document, after the title page.

:ADDRESS—Address

The :ADDRESS and :EADDRESS tags identify the street or mailing address of the author or publisher. The :ALINE tag identifies the individual lines of the address.

:ADDRESS	
:EADDRESS	

:ALINE—Address Line

The :ALINE tag identifies a line of an address. It can appear only within an address element and can be repeated as often as necessary.

:ALINE		.address line
--------	--	---------------

Usage: Address elements can occur within a title page element to give the address or addresses of the author or authors. As many address elements can be specified as necessary, and each address element can contain as many address lines as necessary.

Address elements can also occur anywhere in text.

The address line cannot contain any other tags.

Example

```
:titlep.  
:title stitle='GML Starter Set Reference'.  
Generalized Markup Language Starter Set Reference  
:docnum.SH20-9187-06  
:date.  
:author.Sue Smith  
:address.  
:aline.Information Development XIX  
:aline.Silverado, Colorado  
:eaddress.  
:etitlep.
```

Processing: When an address element appears within a title page element, the information identified is used in creating a title page.

When an address element appears outside a title page element, the lines of the address are formatted as if they were a compact simple list.

Formatting Results: Figure 10 on page 113 illustrates the title page produced by the starter set.

:APPENDIX—Appendix Section

The :APPENDIX tag identifies a major element of a document that contains explanatory and illustrative material helpful to the reader but not essential to the main text.

:APPENDIX	
------------------	--

Usage: The appendix section occurs immediately after the body section. It can contain one or more first-level headings.

The back matter element ends the appendix section.

Example: This document contains:

```
:gdoc.  
:frontm.  
  
:  
:body.  
  
:  
:appendix.  
:h1.GML Messages  
  
:  
:backm.  
  
:  
:egdoc.
```

Processing: Each level-one heading in the appendix section is prefixed with the word "Appendix" and lettered serially, starting with A, whether you have requested numbering with the SYSVAR H option or not. (See "SYSVAR Option" on page 133.) If you have requested heading numbering, heading levels two through four will be lettered and numbered. Following is an example of what a numbering sequence might look like for the first chapter in an appendix section:

```
Head 1  Appendix A  
Head 2  A.1  
Head 3  A.1.1  
Head 2  A.2  
Head 2  A.3  
Head 3  A.3.1  
Head 4  A.3.1.1  
Head 4  A.3.1.2
```

The appendixes are formatted in the same style requested for the body of the document with the SYSVAR S option. The headings that appear within the appendixes are listed in the table of contents.

Formatting Results: This document contains several appendixes, beginning with Appendix A, "The DSMPROF4 Profile" on page 139.

:BACKM—Back Matter Section

The **:BACKM** tag identifies a major element of a document that contains reference material such as a glossary, a bibliography, and an index.

:BACKM	
---------------	--

Usage: The back matter section occurs immediately after the appendix section, or after the body section if there is no appendix section. It can contain an index as well as one or more level-one headings.

Example: This document contains:

```
:gdoc.  
:frontm.  
  
:  
:body.  
  
:  
:appendix.  
  
:  
:backm.  
:h1.Glossary  
  
:  
:index.  
:egdoc.
```

Processing: Automatic head-level numbering is suppressed for headings in the back matter, even if requested with the **SYSVAR H** option. The back matter is always formatted in a two-column page layout, regardless of the value of the **SYSVAR S** option. (See “**SYSVAR Option**” on page 133.)

Formatting Results: This document's back matter begins with “Glossary” on page 161.

:BODY—Body Section

The :BODY tag identifies a major element, the main text of the document.

:BODY	
--------------	--

Usage: The body section occurs immediately after the front matter section. It can contain any of the basic document elements listed in Figure 5 on page 13.

The body section implicitly ends at the appendix or back matter sections.

Example: This document contains:

```
:gdoc.  
:frontm.  
  
:  
:body.  
:h1.Introduction  
  
:  
:appendix.  
  
:  
:backm.  
  
:  
:egdoc.
```

Processing: The pages of the body of the document are identified by Arabic numerals, starting with 1. The layout of the page is determined by the SYSVAR S option, and headings (levels zero through four) are automatically numbered if requested with the SYSVAR H option. (See “SYSVAR Option” on page 133.) Level-zero through level-four headings that appear within the body are listed in the table of contents.

Formatting Results: This page appears within the body of this document, which begins with Chapter 1, “Introduction” on page 1.

:CIT—Title Citation

The :CIT tag identifies the title of a publication, and the :ECIT tag identifies the end of the citation.

:CIT	
:ECIT	

Usage: Citations can occur anywhere in text.

Example:

Both :cit.An End Of Spring:ecit. and
:cit.A Hall Of Mirrors:ecit. were
first novels.

Processing: Title citations are formatted in highlight level one.

Formatting Results: The example previously shown formats as:

Both *An End Of Spring* and *A Hall Of Mirrors* were first novels.

:DL—Definition List

The :DL tag identifies a list of words or phrases and their corresponding definitions, descriptions, or explanations, and the :EDL tag identifies the end of the definition list. The :DTHD tag identifies the definition term heading, whereas the :DDHD tag identifies the definition description heading. The :DT tags and :DD tags, respectively, identify the terms being defined and their descriptions.

:DL	$\left[\text{HEADHI} = \frac{3}{\text{heading-highlight-level}} \right]$ $\left[\text{TERMHI} = \frac{2}{\text{term-highlight-level}} \right]$ $\left[\text{TSIZE} = \frac{10}{\text{term-size}} \right]$ [BREAK] [COMPACT]
:EDL	

Attributes

- HEADHI** identifies the level of highlighting to be used for headings for definition lists. As described in “:HP0 – :HP3—Highlighted Phrases” on page 66, the highlighting level can be a number from 0 to 3, and it defaults to 3 if not entered.
- TERMHI** identifies the level of highlighting to be used for definition terms. As described in “:HP0 – :HP3—Highlighted Phrases” on page 66, the highlighting level can be a number from 0 to 3, and it defaults to 2 if not entered.
- TSIZE** gives the amount of horizontal space to be reserved for the terms and gutter (the space between the terms and descriptions), and it defaults to 10 if not entered.
- BREAK** causes the definition description to start on the line following the definition term if the length of the term exceeds TSIZE. The default is to leave one blank after the term and then begin the definition on the same line when the term exceeds TSIZE.
- COMPACT** indicates that definitions should not be separated from each other by vertical white space. Definitions are ordinarily separated from each other by a blank line.

:DTHD—Definition Term Heading

The :DTHD tag identifies the heading for terms defined in a definition list and is valid only within a definition list.

:DTHD		.definition term heading
-------	--	--------------------------

:DDHD—Definition Description Heading

The **:DDHD** tag identifies the heading for descriptions in a definition list and is valid only within a definition list.

:DDHD		.definition description heading
--------------	--	--

:DT—Definition Term

The **:DT** tag identifies the term or phrase being defined by a definition list item and is valid only within a definition list.

:DT		.definition term
------------	--	-------------------------

:DD—Definition Description

The **:DD** tag identifies the definition of the term or phrase being defined by a definition list item and is valid only within a definition list.

:DD	
------------	--

:LP—List Part

The **:LP** tag identifies a comment or explanation that applies to a part of a list. It can be placed anywhere within a definition list, immediately preceding those items to which it applies.

:LP	
------------	--

Usage: Definition lists can occur anywhere in text; definition lists can be nested within other lists or definition lists, and other lists can be nested within definition lists.

You can specify column headings for definition lists. The **:DTHD** tag identifies the heading for the definition terms column. The **:DDHD** tag identifies the heading for the definition descriptions column. Each of the headings must be entered on a single line in the source document and cannot contain any other tags.

Each item in a definition list is identified by a pair of tags: **:DT** identifies the term being defined, and **:DD** identifies the definition of the term:

- The term being defined must be entered on a single line in the source document and cannot contain any other tags.
- The description of the term is an implied paragraph and can contain any of the text items listed in Figure 4 on page 12. It can also contain any of the basic document elements listed in Figure 5 on page 13, including other lists.

A list part is an implied paragraph and can contain any of the text items listed in Figure 4.

Example: Following is an example of a definition list with headings. We'll change the default for the definition terms (:DT) to highlight level 0 instead of the default of 2, and we'll specify the BREAK value attribute.

```
:dl termhi=0 break.  
:dthd.Animal  
:ddhd.Description  
:dt.coati  
:dd.A tropical American mammal related to the raccoon but with  
a longer body and tail and a long flexible snout  
:dt.fisher  
:dd.A large dark brown North  
American mammal related to  
the weasel  
:dt.margay  
:dd.A small American spotted  
cat resembling the ocelot and  
ranging from southernmost Texas to Brazil  
:lp.The next entry is really  
a bird, which is a class of  
warm-blooded vertebrates  
distinguished by having the body  
more or less completely covered  
with feathers and the forelimbs  
modified as wings.  
:dt.ring-necked pheasant  
:dd.Any of various pheasants  
with white neck rings that have  
been widely introduced in  
temperate regions as game birds  
:edl.
```

Processing: Definition lists are formatted as hanging indent lists; the amount of indentation is given by the TSIZE attribute and defaults to 10. The term is highlighted with the level indicated by the TERMHI attribute, and it defaults to 2. If any of the definition terms exceeds the TSIZE, the BREAK value attribute can be used to force the description to the next line. The default values for HEADHI, TERMHI, and TSIZE attributes can easily be changed. See Appendix A, "The DSMPROF4 Profile" on page 139 for details.

Formatting Results: The example of a definition list shown above formats to look like this:

<i>Animal</i>	<i>Description</i>
coati	A tropical American mammal related to the raccoon but with a longer body and tail and a long flexible snout
fisher	A large dark brown North American animal related to the weasel
margay	A small American spotted cat resembling the ocelot and ranging from south Texas to Brazil

The next entry is really a bird, which is a class of warm-blooded vertebrates distinguished by having the body more or less completely covered with feathers and the forelimbs modified as wings.

ring-necked pheasant

Any of various pheasants with white neck rings that have been widely introduced in temperate regions as game birds

:FIG—Figure

The :FIG tag identifies a diagram, table, or other illustration, and the :EFIG tag identifies the end of the figure. The figure can also contain a figure caption and a figure description, identified by the :FIGCAP and :FIGDESC tags, respectively.

:FIG	[ID = figure-id]
	[FRAME = $\frac{\text{RULE}}{\text{BOX}}$ NONE 'string']
	[PLACE = $\frac{\text{TOP}}{\text{BOTTOM}}$ INLINE]
	[WIDTH = $\frac{\text{PAGE}}{\text{COLUMN}}$ horizontal-width]
	[DEPTH = vertical-depth]
:EFIG	

:FIGCAP—Figure Caption

The :FIGCAP tag identifies a short title, or *caption*, for the figure and is valid only within a figure. A caption for a figure is necessary if the figure is to be numbered for cross-referencing and for inclusion in the list of illustrations.

:FIGCAP	figure caption
---------	----------------

:FIGDESC—Figure Description

The :FIGDESC tag identifies a description of the figure and is valid only within a figure.

:FIGDESC	
----------	--

Attributes

ID gives a unique *identifier* that can be used when referring to the figure. (See ":FIGREF—Figure Reference" on page 53.) The identifier can be up to seven numbers and uppercase and lowercase letters; the identifier is case sensitive. That is, the same letter is a different value in lowercase from uppercase. (A is different from a, B is different from b, and so on.) No other figure can have the same identifier.

If the ID attribute is not entered, no identifier and no figure number are assigned to the figure.

FRAME identifies the manner in which the figure is to be set off from the body of the text. The frame can be a box, a rule (a heavy horizontal line), or a string of characters repeated in place of the rule.

If the FRAME attribute is not entered, a rule is used to separate the figure from the text. The rule appears:

- Before the figure if the figure is placed at the bottom of the page
- After the figure if the figure is placed at the top of the page
- Before and after the figure if the figure is placed inline.

PLACE identifies where the figure is to be placed. The figure can be placed *inline* with the text that surrounds it. In offset style, when you are using both floating and inline figures, the inline figures may be offset, or they can be *float*ed to the top or bottom of the next column or page. The default values for both the PLACE and WIDTH attributes can easily be changed. See Appendix A, "The DSMPROF4 Profile" on page 139 for details.

If the PLACE attribute is not entered, the figure floats to the top of the next column or page.

WIDTH indicates whether the figure will fit within a single column or must extend to full-page width if you are running your document in other than one-column style. Any of the space unit notations illustrated in Table 1 on page 22 can be used.

DEPTH specifies that an amount of vertical white space be included in the figure before the text of the figure. Any of the space unit notations illustrated in Table 1 on page 22 can be used.

If the DEPTH attribute is not entered, the figure is as large as necessary to contain the text between the :FIG and :EFIG tags, up to one full page.

Usage: Figures can occur anywhere in text, except within other figures, tables, examples, or footnotes. The body of a figure is composed of all the text between the :FIG tag and the :EFIG tag. It can contain any basic document elements (except other figures, tables, examples, or footnotes—even though references to figures, tables, examples, and footnotes can be included), lines of text, character graphics, control words, and macros.

Figures must be either column wide or page wide. The size you indicate on the WIDTH attribute simply indicates whether the figure will fit in the column or needs to extend the full width of the page. Specifying space units rather than PAGE or COLUMN for the WIDTH attribute allows you to create figures that are somewhat independent of the general layout of the document. If you have to print the document on different devices that have different column or page widths, or if you decide to change a single-column document to two columns, you will not have to change the markup for the figure.

A figure can also contain a figure caption, identified by the :FIGCAP tag, placed at the bottom of the figure. The caption must occupy a single line in the source document and cannot contain any other tags.

Further descriptive text following the figure caption is identified with the :FIGDESC tag. It can contain any basic document elements (except other figures, tables, examples, or footnotes—even though references to figures, tables, examples, and footnotes can be included), lines of text, character

graphics, control words, and macros. The description can continue for more than one line and is ended by the :EFIG tag.

Example: Markup for an inline illustration surrounded by a box looks like this:

```
:fig place=inline frame=box.  
  
:  
:figcap.Illustration  
:efig.
```

A checklist to be placed at the bottom of the next available column can be marked up as:

```
:fig place=bottom id=chk.  
  
:  
:figcap.Checklist  
:figdesc.Ensure that all hatches  
are closed before submerging.  
:efig.
```

Be aware that our formatted figure examples do not look the same as when used with an unmodified GML starter set.

The latter figure has been given a unique identifier of "chk"; this identifier can be used when referring to the figure elsewhere in the document with a :FIGREF tag (provided the figure has a caption).

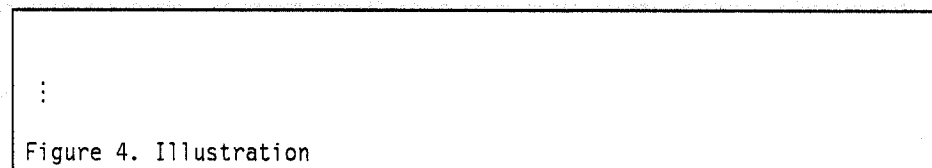
Processing: The normal formatting of text (concatenation and justification) is suspended within a figure; each input line becomes a separate output line. Other GML tags should be used with caution within figures, because concatenation of input lines is suppressed. Unexpected line breaks or concatenation of source text lines can occur when GML tags are used within figures. Neither hyphenation nor spelling verification is performed.

The formatting environment, which includes such things as indentation, centering, and character translation, is saved at the beginning of a figure and restored at the end of a figure. (In this document, the amount of white space preceding figures is one line, and the indentation is 2. These values are set in the profile of the document. See Appendix A, "The DSMPROF4 Profile" on page 139 for details.) For more information on the active formatting environment, refer to the *Document Composition Facility: SCRIPT/VS Text Programmer's Guide*.

When a figure caption is specified with the :FIGCAP tag, the figure is automatically assigned a figure number. The figure number is included in the formatted figure caption and, along with the caption itself, in the List of Illustrations that can be placed in the front matter of the document by including the :FIGLIST tag. If no figure caption is included, the figure is not numbered and will not appear in the List of Illustrations. (See ":FIGLIST—List of Illustrations" on page 52.)

Note: Processing may be incorrect when text is intermixed with GML tags inside of figures.

Formatting Results: The first example formats as:



The second example formats as:

Figure 5. Checklist: Ensure that all hatches are closed before submerging.

:FIGLIST—List of Illustrations

The :FIGLIST tag identifies the place in the document where a list of illustrations will be created.

:FIGLIST	
-----------------	--

Usage: The list of illustrations normally is placed in the front matter of a document. If the :FIGLIST tag is not entered, a list of illustrations will not be created.

Example: This document contains:

```
:gdoc.  
:frontm.  
  
:  
:toc.  
:figlist.  
:body.  
  
:
```

Processing: The list of illustrations is placed on a new page, preceded by the heading "List of Illustrations." If duplexing has been requested via the SYSVAR D option, the list will be placed on an odd numbered page (See "SYSVAR Option" on page 133.) Those figures in the document that have figure captions, identified with the :FIGCAP tag, are listed by figure number, along with the figure caption and page number.

When the :FIGLIST tag is placed in the front matter, the list of illustrations will be empty unless two or more formatting passes are specified with the FPASSES or TWOPASS option of the SCRIPT command. (See "SCRIPT Command Options" on page 117.) The list of illustrations will be empty because the entries in the list of illustrations are references to figures defined later in the document. Before final formatting of a document for publication, you may want to place the :FIGLIST tag temporarily in the back matter so you can run the book with a single pass. If you place your :FIGLIST in the back matter, it will format in two-column style.

Formatting Results: A figure list is formatted.

Note: Placement of the :FIGLIST tag in other than the front matter or back of the document can produce unexpected results.

:FIGREF—Figure Reference

The :FIGREF tag identifies a place in a document where a reference to a captioned figure should be made.

:FIGREF	REFID = figure-id [PAGE = YES NO]
----------------	---

Attributes

REFID identifies the figure being referred to. The value must be a unique identifier assigned to a figure with the ID attribute of the :FIG tag. The REFID is case sensitive and must be identical to the ID to which it refers.

PAGE indicates whether the figure's page number should or should not be included in the reference. If the PAGE attribute is not entered, the figure's page number is included in the reference only if the figure and reference are on different pages.

Usage: References to uniquely identified figures can occur anywhere in text.

Example: The unique identifier of Figure 5 on page 13 is bde. This reference to that figure is entered as:

... the unique identifier
of :figref refid=bde. is bde ...

Processing: Figures must be captioned to resolve references correctly. It is the caption that assigns a unique figure number to a figure.

Specify two or more formatting passes with the FPASSES or TWOPASS option of the SCRIPT command to correctly resolve forward references and to correctly include and resolve the "on page x" phrase. (See "SCRIPT Command Options" on page 117.)

Note: Do not end an input line with this tag. Ending an input line with this tag may cause the next input line to be concatenated to the text inserted by the tag without an intervening blank.

Formatting Results: The reference to the figure whose identifier is 'bde' formats as:

... the unique identifier of Figure 5 on page 13 is bde ...

:FN—Footnote

The :FN tag identifies a note of reference, explanation, or comment to be placed below the text on the printed page. The :EFN tag identifies the end of the footnote.

:FN	[id = footnote-id]
:EFN	

Attributes

ID gives a unique *identifier* that can be used when referring to the footnote. (See ":FNREF—Footnote Reference" on page 56.) The identifier can be up to seven numbers and uppercase and lowercase letters; the identifier is case sensitive. No other footnote can have the same identifier.

If the ID attribute is not entered, no identifier is assigned to the footnote, and the footnote is referenced where it is defined.

Usage: Footnotes can occur anywhere in text, except within other footnotes, figures, examples, headings, or tables. To ensure that the footnote and the reference to the footnote appear on the same page, they should be entered in the source document as closely together as possible when using the ID attribute. The ID attribute is useful only when you want to refer to the same footnote from more than one place within your document, or when you want to refer to a footnote within an example or figure. The body of a footnote is composed of all the text between the :FN tag and the :EFN tag; it can contain any basic document elements except other footnotes, figures, or examples.

Example:

```
:p.  
The entire footnote  
is placed at the bottom of the  
current page if it will fit.  
:fn.  
If at least two lines of the  
footnote will fit on the current  
page, the footnote is started on  
the current page and the rest of  
the footnote is placed at the bottom  
of the next available page.  
:efn.
```

Processing: The text of the footnote and a footnote number are placed at the bottom of the current page if there is room. If only one line of the footnote will fit on the current page, the footnote is placed on the next available page. If at least two lines of the footnote fit on the current page, the footnote will be split; as many lines as possible are placed on the current page and the rest of the footnote continues on the next available page.

If no ID attribute is given with the :FN tag, a footnote reference is inserted into the text where the footnote is defined. If an ID attribute is given, the location of the footnote reference must be given with the :FNREF tag. If you give the footnote an ID, the footnote still appears on the page where the footnote is defined, regardless of where in the document the footnote reference (:FNREF) is placed.

Formatting Results: The example given above formats as:

The entire footnote is placed at the bottom of the current page if it will fit.⁵

⁵ If at least two lines of the footnote will fit on the current page, the footnote is started on the current page and the rest of the footnote is placed at the bottom of the next available page.

:FNREF—Footnote Reference

The :FNREF tag identifies a place in a document where a reference to a footnote should be made.

:FNREF	REFID = footnote-id
---------------	---------------------

Attributes

REFID identifies the footnote being referred to. The value must be a unique identifier assigned to a footnote with the ID attribute of the :FN tag. The REFID is case sensitive and must be identical to the ID to which it refers.

Usage: References to uniquely identified footnotes can occur anywhere in text. To ensure that the footnote and the reference to the footnote appear on the same page, they should be entered in the source document as closely together as possible when using the ID attribute.

You can use the :FNREF tag to refer to the same footnote as many times as you need to. The footnote itself, however, appears only on the page where it was defined (or the next page if it will not fit on the current page).

Example:

```
:p.The entire footnote
:fn id=oflow.
If at least two lines of the
footnote will fit on the current
page, the footnote is started on
the current page and the rest of
the footnote is placed at the
bottom of the next available page.
:efn.
is placed at the bottom of the
current page if it will
fit.:fnref refid=oflow.
```

Processing: The footnote reference is printed in superscript numbers on devices that have such numbers. Otherwise, the reference is enclosed in parentheses.

If the footnote reference precedes the footnote in the source document, you must specify two or more formatting passes with the FPASSES or TWOPASS option of the SCRIPT command. (See "SCRIPT Command Options" on page 117.)

Formatting Results: The example given above formats as:

The entire footnote is placed at the bottom of the current page if it will fit.⁶

⁶ If at least two lines of the footnote will fit on the current page, the footnote is started on the current page and the rest of the footnote is placed at the bottom of the next available page.

:FRONTM—Front Matter Section

The **:FRONTM** tag identifies a major element of a document that contains material that serves as a guide to the document's contents and nature, such as the abstract, preface, table of contents, and list of illustrations.

:FRONTM	
----------------	--

Usage: The front matter section is usually the first section of a document and contains standard information in a standard order. Not all of these elements are always present in the front matter, but when they are, they usually appear in this order:

- title page
- abstract
- preface
- table of contents
- list of illustrations
- list of tables

The front matter section is implicitly ended by the body section.

Example: This document contains:

```
:gdoc.  
:frontm.  
:titlep.  
:  
:etitlep.  
:preface.  
:  
:toc.  
:figlist.  
:tlist.  
:body.  
:  
:egdoc.
```

Processing: Pages in the front matter are numbered in Roman numerals, starting from i (the page number is suppressed on the title page). All the material that appears in the front matter is in single-column format. The front-matter headings will not appear in the table of contents nor will they be numbered. Automatic head-level numbering is suppressed for headings in the front matter, even if requested with the **SYSVAR H** option. (See “**SYSVAR Option**” on page 133.)

Processing Results: This document's front matter precedes Chapter 1, “Introduction” on page 1.

:GDOC — General Document

The :GDOC tag identifies a general document, and the :EGDOC tag identifies the end of a general document.

:GDOC	SEC = 'security classification' LANGUAGE = 'language name'
:EGDOC	

Attributes

SEC identifies the security classification of the document. The value can be any character string, but if the text of the security classification contains any blanks or special characters, it must be enclosed in single quotation marks ('').

If the SEC attribute is not entered, no security classification is used.

LANGUAGE selects the language to be used for the translation of literals and messages, and for spelling verification, hyphenation, and index sorting sequences in 15 different languages. The following list shows the settings that are performed:

- Translation of all profile literals
- Translation of all profile GML messages
- Dictionary used for spelling verification
- Dictionary and algorithm used for hyphenation
- Index sort sequence used
- Uppercase translation
- Full-stop characters
- Specification of quotation marks used with :Q tag.
- Hyphenation of headings

The LANGUAGE attribute does not constitute National Language Support enablement for the starter set.

The default language for the GML starter set is the language that was selected at installation time. You can query &\$\$VLG to find out what the default language is at your installation. If you do not specify the LANGUAGE attribute, you will get the default.

Only one GDOC tag with the language attribute can be specified for each document.

The following list shows the languages that can be specified on the language attribute:

Value	Language
CENGLISH	Canadian English
CFRENCH	Canadian French

DANISH	Danish
DUTCH	Dutch
ENGLISH	American English
FINNISH	Finnish
FRENCH	French National
GERMAN	German
ICELANDIC	Icelandic
ITALIAN	Italian
NORWEGIAN	Norwegian
PORTUGUESE	Portuguese
SPANISH	Spanish
SWEDISH	Swedish
UKENGLISH	United Kingdom English

Usage: The :GDOC tag is usually the first tag in a document; the :EGDOC tag is usually the last tag in a document.

Example: This document contains:

```
:gdoc language=english.
:frontm.
:titlep.
...
:etitlep.
...
:egdoc.
```

A very sensitive document might contain:

```
:gdoc sec='Very Sensitive Information'.
...
:egdoc.
```

Processing: The security classification is centered at the top of every printed page as a level-two highlighted phrase. The security classification also appears on the title page but is not highlighted.

Formatting Results: If the SEC attribute is specified on the GDOC tag, DSMPROF4 formats the security line as a running heading. If the LANGUAGE attribute is specified on the GDOC tag, DSMPROF4 formats the literals, messages, and so on, in the language specified.

:GL—Glossary List

The :GL tag identifies a list of glossary entries and their corresponding definitions, and the :EGL tag identifies the end of the glossary list. The :GT tags and :GD tags, respectively, identify the terms being defined and their definitions.

:GL	[COMPACT] [TERMHI = $\frac{2}{\text{term-highlight-level}}$]
:EGL	

:GT—Glossary Term

The :GT tag identifies the term or phrase being defined as a glossary entry and is valid only within a glossary list.

:GT		glossary term
-----	--	---------------

:GD—Glossary Definition

The :GD tag identifies the definition of the glossary entry being defined and is valid only within a glossary list.

:GD	
-----	--

:LP—List Part

The :LP tag identifies a comment or explanation that applies to a part of a list. It can be placed anywhere within a glossary list, immediately preceding those items to which it applies.

:LP	
-----	--

Attributes

COMPACT

indicates that glossary terms should not be separated from each other by vertical white space. Items are ordinarily separated from each other by a blank line.

TERMHI

identifies the level of highlighting to be used for glossary terms. As described in “:HP0 – :HP3—Highlighted Phrases” on page 66, the highlighting level can be a number from 0 to 3, and it defaults to 2 if not entered.

Note: The default value for the TERMHI attribute can easily be changed. See Appendix A, “The DSMPROF4 Profile” on page 139 for details.

Usage: A glossary list can be used in the back matter of a document to format a glossary.

Each glossary entry is identified by a pair of tags. :GT identifies the term being defined, and :GD identifies the definition of the term:

- The term being defined must be entered on a single line in the source document and cannot contain any other tags.
- The definition of the term is an implied paragraph and can contain any of the text items listed in Figure 4 on page 12.

Example: A few of the entries in this document's glossary are marked up as:

```
:gl.  
:  
:gt.alphanumeric string  
:gd.A sequence of characters  
consisting solely of the letters  
a through z and the numerals  
0 through 9.  
:gt.ampersand  
:gd.The & character.  
:lp.When an ampersand begins a  
character string, SCRIPT/VS  
assumes the character string  
is a symbol name.  
If the symbol name is defined,  
SCRIPT/VS replaces the symbol  
with its value (unless symbol  
substitution is off).  
:  
:egl.
```

Processing: Glossary lists are formatted by highlighting the term as a level-two highlighted phrase and by separating the term from the definition by a colon.

Formatting Results: The glossary entries previously given are in this document's glossary, which follows the appendixes.

:H0 – :H6—Headings

The :H0 and :H1 tags identify major sections of a document. Level-zero sections are sometimes called *parts*. Level-one sections are called *chapters* when they appear in the body of the document, and *appendixes* when they appear in the appendix section.

:H0, :H1	[ID = heading-id] [STITLE = 'short heading']	.section heading
----------	---	------------------

The :H2 through :H6 tags identify subsections of a document, and they are sometimes called *topics*.

:H2, :H3, :H4, :H5, :H6	[ID = heading-id]	.heading
-------------------------	-------------------	----------

Attributes

ID gives a unique *identifier* that can be used when referring to the heading. (See “:HDREF—Heading Reference” on page 65.) The identifier can be up to seven numbers and uppercase and lowercase letters; the identifier is case sensitive. No other heading can have the same identifier.

If the ID attribute is not entered, no identifier is assigned to the heading.

STITLE gives a short version of the heading to be used in place of the actual heading text at the bottom of the page as a running footing. The value can be any text, but if the text of the short heading contains blanks or special characters, it must be enclosed in single quotation marks ('').

If the STITLE attribute is not entered, the full text of the heading is used in the running footing.

Note: The ampersand (&) character should not be used in the STITLE text of the :H0 and :H1 tags. Also, when the STITLE attribute is not specified, the ampersand should not be used in the section heading text. The ampersand is the symbol character for the page number, and its use in a :H0 or :H1 tag will result in the current page number being printed in the text of the running footing. If an ampersand must be used in the text of a head-level control, use the & symbol.

Usage: Headings are used throughout a document to subdivide it into sections and to illuminate the document's overall structure. Heading sections can contain any of the basic document elements listed in Figure 5 on page 13, as well as lower-level headings.

The text of a heading should be entered with initial capital letters, because headings appear in the table of contents as you have entered them, regardless of how the heading is formatted in the document itself.

The text of a heading must be entered on a single line of the source document (either all on the same line as the tag markup or all on the line following the tag markup) and cannot contain any tags.

Function	:H0	:H1	:H2	:H3	:H4	:H5	:H6
Begins a new page	yes	yes					
Heading inline with text						yes	yes
Line device highlight level	3	3	3	3	2	2	1
Line device heading capitalization	yes	yes	yes				
Page device font size	20	20	18	14	12	10	10
Page device (B)old/(I)talic font	BI	B	BI	B	BI	B	BI
Heading numbered		yes	yes	yes	yes		
Table of Contents entry	yes	yes	yes	yes			
ID attribute recognized	yes	yes	yes	yes	yes	yes	yes
STITLE attribute recognized	yes	yes					

Table 2. Summary of Head-Level Tags. If the SYSVAR D option is specified, the new page started by :H0 and :H1 is odd-numbered, and the :H0 and :H1 headings are right justified for one- and two-column layouts. :H1 through :H4 (in the body and appendix sections) are numbered only if numbering is requested with the SYSVAR H option. (See "SYSVAR Option" on page 133.) Headings that appear in the front matter are not listed in the table of contents and are never numbered.

Example: This chapter begins with

```
:h1 id=tags.GML Starter Set Tags
:p.The following pages ...
```

which could also have been entered like this:

```
:h1 id=tags.
GML Starter Set Tags
:p.The following pages ...
```

This section begins with:

```
:h5.Example
:p.This chapter begins ...
```

Processing: Headings are handled differently depending on:

- Whether you have requested duplexing with the SYSVAR D option
- Whether your headings appear in the front matter (:FRONTM), the body (:BODY), the appendix (:APPENDIX), or the back matter (:BACKM)
- Whether you have requested offset style with the SYSVAR S option.

Without duplexing, the level-zero and level-one headings:

- Are left justified
- Start at the top of the next available page
- Provide the text (or the shortened version given with the STITLE attribute) for running footings on both the odd- and even-numbered pages. The page number is aligned at the outside margin, and the text is aligned at the inside margin.

With duplexing, the level-zero and level-one headings:

- Are, with the exception of offset style, right justified
- Start at the top of the next odd-numbered page
- Provide the text (or the shortened version given with the STITLE attribute) for running footings on the odd-numbered pages. If you have a title page, the title as specified on the :TITLE tag (or the shortened version given with the STITLE attribute) appears in the running footing on the even-numbered pages. The page number and text of the running footing are both at the outside margin rather than being split across the bottom of the page. If you do not have a title page, *only* the page number appears as the running footing on even-numbered pages.

Headings in the front matter (:FRONTM):

- Are not listed in the table of contents
- Are not numbered, even if numbering is requested on the SYSVAR H option of the SCRIPT command.

Headings (level zero through four) in the body (:BODY):

- Are listed in the table of contents
- Are numbered if requested on the SYSVAR H option of the SCRIPT command.

Headings (level zero through four) in the appendix section (:APPENDIX):

- Are listed in the table of contents
- Are numbered (level two through four) if requested on the SYSVAR H option of the SCRIPT command
- Are prefixed with the word Appendix and lettered serially beginning with the letter A. (See ":APPENDIX—Appendix Section" on page 39 for an example of the numbering sequence.)

Headings (level zero through four) in the back matter (:BACKM):

- Are listed in the table of contents
- Are not numbered, even if numbering is requested on the SYSVAR H option of the SCRIPT command.

Headings in offset style cause section breaks. This means that the first few lines of text following the heading might not be kept on the same page as the heading itself.

Heading levels 5 and 6 are never included in the table of contents nor are they ever numbered.

Formatting Results: The text Chapter 2, "Marking Up General Documents" on page 11 is a level-one heading, and the text "Usage" on page 62 is a level-five heading.

:HREF—Heading Reference

The :HREF tag identifies a place in a document where a reference to a heading should be made.

:HREF	REFID = heading-id [PAGE = YES] [PAGE = NO]
-------	---

Attributes

REFID identifies the heading being referred to. The value must be a unique identifier assigned to a heading with the ID attribute of the :Hn tag. The REFID is case sensitive and must be identical to the ID to which it refers.

PAGE indicates whether the page number of the heading should be included in the reference. YES means to add the page number even if the heading is on the same page as the reference. NO means do not add the page number even if the heading is on a different page. If the PAGE attribute is not entered, the page number of the heading is included in the reference only if the heading and reference are on different pages.

Usage: References to uniquely identified headings can occur anywhere in the text.

Example: To refer to the heading shown as an example in “:H0—:H6—Headings” on page 62, enter:

... See :hhref refid=tags. for details. ...

If you didn't want the page number to appear as a part of the reference, you would add the PAGE attribute, like this:

... See :hhref refid=tags page=no. for details. ...

Processing: Specify two or more formatting passes with the FPASSES or TWOPASS option of the SCRIPT command to correctly resolve forward references and to correctly include and resolve the “on page x” phrase. (See “SCRIPT Command Options” on page 117.)

Do not end an input line with this tag. Ending an input line with this tag might cause the next input line to be concatenated to the text inserted by the tag, without an intervening blank.

Formatting Results: The first example previously shown formats as:

... See Chapter 3, “GML Starter Set Tags” on page 35 for details. ...

The example using the PAGE attribute formats as:

... See Chapter 3, “GML Starter Set Tags” for details. ...

:HP0 – :HP3—Highlighted Phrases

The :HP0, :HP1, :HP2, and :HP3 tags identify a word or phrase that is to be emphasized. The :EHP0, :EHP1, :EHP2, and :EHP3 tags identify the ends of highlighted phrases.

:HP0, :HP1, :HP2, :HP3	
:EHP0, :EHP1, :EHP2, :EHP3	

Usage: Highlighted phrases can occur anywhere in the text to indicate the degree of emphasis to be given a particular phrase and can be nested within other highlighted phrases. :HP0 corresponds to the style of text used in the body of the document and is normally used only to de-emphasize part of a highlighted phrase.

A highlighted phrase can be an entire sentence, a paragraph, or only part of a word.

Example:

```
... :hp2.All:ehp2. :hp1.of
this sentence is
high:hp3.light:ehp3.ed:ehp1..
Be careful when placing
the high:hp3.
light
:ehp3.ed
phrase tags in your text or you :hp1.will:ehp1. get
unexpected spacing results. ...
```

The highlight levels used in this book are defined as follows:

LEVEL	DESCRIPTION
0	No Emphasis
1	<i>Italicized</i>
2	Bold
3	<i>Italicized and Bold</i>

Highlight levels vary, depending on the output device used. The following table lists the default highlight levels for different devices.

Logical Devices	HP1	HP2	HP3
1403, 2741, STAIRS	underscore	overstrike	overstrike and underscore
3270, 3800 ⁷	underscore	uppercase	uppercase and underscore
PG1, 3820, 38PP, 4250, PS, PG4	italic	bold	bold italic
PG2, PG3	italic	bold	bold uppercase

Processing: The method used for highlighting depends on the device used for printing. On page devices and PostScript devices, bold, italic, and bold italic versions of the current font are used. Other devices use underscoring, capitalization, overstriking, or a combination of them.

Formatting Results: The example given above formats as:

... All of this sentence is *highlighted*. Be careful when placing the high *light* ed phrase tags in your text or you *will* get unexpected spacing results. ...

⁷ The 3800 device will use the &\$CHAR fonts 2-4 (if they are specified) for highlight levels 1-3 respectively.

:I1, :I2, and :I3—Index Entries

The :I1 tag identifies a primary index term. All primary index terms are collected and placed alphabetically in the index.

The :I2 tag identifies a secondary index term. All secondary index terms with the same primary term are collected and placed after the primary term. Unless you specify otherwise, the :I2 entries are placed beneath the most recently specified primary term.

The :I3 tag identifies a tertiary index term. All tertiary terms with the same primary and secondary terms are collected and placed after the secondary term. Unless you specify otherwise, the :I3 entries are placed beneath the most recently specified primary and secondary terms.

Note: These tags are ignored if the INDEX option of the SCRIPT command is not used. You must, additionally, include the :INDEX tag in your document at the point where you want the index to be printed.

:I1, :I2, :I3	[ID = index-id] [START END PG = MAJOR 'string'] [REFID = index-id]	.index term
---------------	---	-------------

Attributes

ID gives a unique *identifier* that can be used when referring to the index entry. (See “:IREF—Index Entry Reference” on page 75.) The identifier can be up to seven numbers and uppercase and lowercase letters; the identifier is case sensitive. No other index entry can have the same identifier.

If the ID attribute is not entered, no identifier is assigned to the index entry.

PG indicates the type of page reference the index term is to include when it appears in the index. START and END can be used to delimit page ranges. MAJOR can be used to indicate the principal page reference for the term. The MAJOR page reference is the first page number listed after the index entry, out of sequence if necessary. The 'string' can identify a string of characters (text) that is to be used in place of a page number; it must be in single quotes.

If PG is not specified, a reference to the page on which surrounding text appears is used.

REFID indicates that the primary and secondary (if specified with :I3 tags) terms of the referenced index entry are to be used in place of the most recently specified primary and secondary terms. The REFID is case sensitive and must duplicate the ID to which it refers.

The REFID attribute is recognized only by the :I2 and :I3 tags.

If the value of the REFID attribute has not been defined, the REFID attribute is ignored, and the index entry is placed under the most recently defined index term. If the REFID attribute on a :I3 tag refers to an identifier on a :I1 tag, the refid attribute is ignored, and the

index entry is placed under the most recently defined secondary index term.

Usage: Index entries can occur anywhere in a document, preceding the index. To ensure the accuracy of page references, place index entries as close to the referenced text as possible in the source document.

The text of an index term must be entered on a single line of the source document and cannot include any other tags.

Example:

```
:i1.weasels
...
:i1.walrus
```

creates two entries in the W section of the index.

Secondary and tertiary index terms are normally subentries of the most recently entered primary and secondary terms, respectively. For example,

```
:i1.weasels
:i2.stroking
:i2 id=care.care and feeding
```

creates two subentries of the primary term "weasels." The index entry "weasels, care and feeding" is assigned the identifier "care."

The index entry "care and feeding" has been given an identifier of "care" with an ID attribute. It is, therefore, possible to refer to that identifier (care) by using the REFID attribute on a :I3 tag elsewhere in the document. It is not necessary to repeat the :I1 and :I2 entries it should follow.

When a secondary or tertiary index term refers to another entry with the REFID attribute, the higher-level terms for the entry are taken from the referenced term. For example,

```
:i1.water fowl
...
:i3 refid=care.dental hygiene
```

creates two entries in the W section: "water fowl" and "weasels, care and feeding, dental hygiene."

We have used the identifier "care" on the REFID with the :I3 index entry "dental hygiene." The "refid=care" refers back to "id=care" and places "dental hygiene" under "care and feeding" that follows "weasels" as shown under "Formatting Results" on this page.

The references following a term in the index can include things other than page numbers. For example,

```
:i1 pg='Plates V-VII'.water fowl
```

When the discussion of a topic covers many pages, the extent of that discussion can be indicated in the index by showing the range of pages it covers:

```
:i1 pg=start.walrus
...
:i1 pg=end.walrus
```

Processing: Terms are sorted alphabetically and formatted when the :INDEX tag is processed. Duplicated terms are formatted as a single index entry with all the page references listed.

SCRIPT/VS ignores the case of index terms when sorting and checking for duplicates. The text of the index term printed in the index is taken from the first occurrence; duplicates contribute only additional page references. For example,

```
:il.weasels
...
:il.WEASELS
```

results in a single entry in the W section of "weasels" with two page references. Blanks, apostrophes, and hyphens in index terms are sorted before letters and numbers. Special characters are sorted at the end of the index following letters and numbers. The sorting can be modified by the IXI and IXB parameters of the SCRIPT/VS .DC control word. See "Controlling Index Sorting (.DC)" on page 28 and the *Document Composition Facility: SCRIPT/VS Language Reference* for additional details. Refer to the *Document Composition Facility: SCRIPT/VS User's Guide* for specific sort sequence information.

Formatting Results: The index terms specified previously might result in the following index:

W

```
walrus 42, 121-125
water fowl 93, Plates V-VII
weasels 29, 33
    care and feeding 33
    dental hygiene 93
    stroking 33
```

:IH1, :IH2, and :IH3—Index Headers

The :IH1 tag identifies a primary index term to be placed in the index with a null page reference. All primary index terms are collected and placed alphabetically in the index.

The :IH2 tag identifies a secondary index term to be placed in the index with a null page reference. All secondary index terms with the same primary term are collected and placed after the primary term. Unless you specify otherwise, the :IH2 entries are placed beneath the most recently specified primary term.

The :IH3 tag identifies a tertiary index term to be placed in the index with a null page reference. All tertiary terms with the same primary and secondary terms are collected and placed after the secondary term. Unless you specify otherwise, the :IH3 entries are placed beneath the most recently specified primary and secondary terms.

Note: These tags are ignored if the INDEX option of the SCRIPT command is not used.

:IH1, :IH2, :IH3	[ID = index-id] [PRINT = 'index entry'] [SEE = 'reference'] [SEEDID = index-id]	.index term
------------------	--	-------------

Attributes

ID gives a unique *identifier* that can be used when referring to the index entry. (See “:IREF—Index Entry Reference” on page 75.) The identifier can be up to seven numbers and uppercase and lowercase letters; the identifier is case sensitive. No other index entry can have the same identifier.

If the ID attribute is not entered, no identifier is assigned to the index entry.

PRINT gives an alternative entry to be printed in the index in place of the index term. The index entry is still sorted based on the term; only the text printed in the index is changed. The value can be any string, but if the text contains blanks or special characters, it must be enclosed in single quotation marks (').

If PRINT is not entered, the index term is printed in the index.

SEE indicates that instead of a page reference, this index term should include a reference to another index term. The value of the SEE attribute is taken to be that term. The value can be any string, but if the text contains blanks or special characters, it must be enclosed in single quotation marks (').

The SEE attribute is recognized only when the referenced term is a primary or secondary index term.

SEEDID indicates that instead of a page reference, this index term should include a reference to another index term. The value must be a unique identifier assigned to an index term with the ID attribute. The identified term's primary index term is taken to be the reference.

The SEEID attribute is recognized when the referenced term is a primary or secondary index term.

Usage: Index headers can occur anywhere in a document, preceding the index. To ensure the accuracy of page references, place tags as close to the referenced text as possible in the source document.

The text of an index term must be entered on a single line of the source document and cannot include any other tags.

Index headers are most often used in conjunction with secondary and tertiary index terms.

Example:

```
:ih1.pines
:i2.Japanese Black
:i2.Lodgepole
```

Index headers are also used to indicate cross-references within the index. For example,

```
:ih1 see=martens.weasels
```

When an alternative index entry is specified with the PRINT attribute, that string is printed in the index, but the entry is still sorted on the basis of the index term. For example,

```
:ih1
print='The Walrus and The Carpenter'.
walrus
```

places the entry "The Walrus and The Carpenter" in the W section of the index, not the T section. Duplicate terms are still recognized on the basis of the index term:

```
:i1.walrus
```

and contribute a page reference to the entry "The Walrus and The Carpenter" in the W section.

Processing: Terms are sorted alphabetically and formatted when the :INDEX tag is processed. Duplicated terms are formatted as a single index entry with all the page references listed.

SCRIPT/VS ignores the case of index terms when sorting and checking for duplicates, and the text of the index term printed in the index is taken from the first occurrence. For example,

```
:ih1.water fowl
...
:ih1.Water Fowl
```

result in a single header of "water fowl" in the W section of the index. Similarly,

```
:ih1 print='IBM 3800'.3800
...
:ih1 print='Nonimpact printer'.3800
```

result in a single entry of "IBM 3800" in the 3 section of the index.

Blanks, apostrophes, and hyphens are sorted ahead of letters and numbers. Special characters are sorted at the end of the index following letters and numbers. The sorting can be modified by the IXI and IXB parameters of the SCRIPT/VS .DC control word. See "Controlling Index Sorting (.DC)" on page 28 and refer to the *Document Composition Facility: SCRIPT/VS Language Reference* for additional details.

Formatting Results: The index terms previously specified might result in the following index:

P

pin

Japanese Black 14

Lodgepole 11

W

The Walrus and The Carpenter 4, 9

water fowl

weasels

See martens

3

IBM 3800

Nonimpact printer

Note: The example above shows how page devices and PostScript devices treat the index entry headings (P, W, and 3). An index formatted for line devices (1403 or IBM 3800 Printing Subsystem Model 1) would have boxes surrounding these headings.

:INDEX—Index

The :INDEX tag identifies a place in the document where an index should be created.

:INDEX	
---------------	--

Usage: The :INDEX tag should be placed immediately before the :EGDOC tag. If the :INDEX tag is not entered, an index is not created.

Example:: This document contains:

```
⋮  
:backm.
```

```
⋮  
:index.  
:egdoc.
```

Processing: The index is placed on a new page, preceded by the heading "INDEX." The index will be empty if the INDEX option of the SCRIPT command is not used. The index is formatted in two-column style, regardless of whether two-column format was requested with the SYSVAR S option.

Running headings and footings will be suppressed on pages following the index.

If duplexing is requested with the SYSVAR D option, the first page of the index will be odd-numbered. (See "SYSVAR Option" on page 133.) The index is composed of index entries identified by the :I1, :I2, :I3, :IH1, :IH2, :IH3, and :IREF tags.

Formatting Results: Examples of indexes are shown under ":IH1, :IH2, and :IH3—Index Headers" on page 71 and ":IREF—Index Entry Reference" on page 75. The index at the back of this book was created using the indexing tags.

:IREF—Index Entry Reference

The :IREF tag identifies a place in a document at which an index entry similar to one already made should also be made.

:IREF	<div data-bbox="743 394 964 422">REFID = index-id</div> <div data-bbox="748 449 976 569">PG = <div data-bbox="846 449 954 485">START</div><div data-bbox="846 485 915 512">END</div><div data-bbox="846 512 954 539">MAJOR</div><div data-bbox="846 539 938 569">'string'</div></div> <div data-bbox="748 583 1052 611">[SEE = 'index reference']</div> <div data-bbox="748 638 980 667">[SEEID = index-id]</div>
-------	---

Attributes

REFID indicates the index entry that is to be duplicated. The value must be a unique identifier assigned to a heading with the ID attribute of the :I1, :I2, or :I3 tag.

PG indicates the type of page reference the index term is to include when it appears in the index. START and END can be used to delimit page ranges. MAJOR can be used to indicate the principal page reference for the term. The 'string' can identify a string of characters (text) that is to be used in place of a page number; it must be in single quotes.

If PG is not specified, a reference to the page on which surrounding text appears is used.

SEE indicates that instead of a page reference, this index term should include a reference to another index term. The value of the SEE attribute is taken to be that term. The value can be any string, but if the text contains blanks or special characters, it must be enclosed in single quotation marks (').

The SEE attribute is recognized only when the referenced term is a primary or secondary index term.

SEEID indicates that instead of a page reference, this index term should include a reference to another index term. The value must be a unique identifier assigned to an index term with the ID attribute. The identified term's primary index term is taken to be the reference.

The SEEID attribute is recognized only when the referenced term is a primary or secondary index term.

Usage: References to uniquely identified index entries can occur anywhere in a document, preceding the index. To ensure the accuracy of page number references, place tags as close to the referenced text as possible in the source document.

An index reference can be used to repeat an index entry specified elsewhere in the document with the ID attribute.

Example:

```

:ih1.weasels
:i2 id=care.care and feeding
...
:iref refid=care.
...
:iref refid=care.

```

will create the entry "weasels, care and feeding" with three page references.

Index references are especially useful in delimiting page ranges. For example,

```

:ih1.pines
:i2 id=bpine pg=start.Japanese Black
...
:iref refid=bpine pg=end.

```

Cross-references within the index can be created to indicate related topics. For example,

```

:i1 id=cones.conifers
...
:iref refid=bpine seeid=cones.
...
:iref refid=cones see=pines.

```

The SEEID attribute takes the value of the ID attribute as its value (for example, id=cones seeid=cones). The SEE attribute takes the actual text of an index entry as its value (for example, if an entry were, :ih1.pines, the reference to it would be see=pines).

Processing: Terms are sorted alphabetically and formatted when the :INDEX tag is processed. Repeated terms are formatted as a single index entry with multiple page references.

SCRIPT/VS ignores the case of index terms when sorting and checking for duplicates. The text of the index term printed in the index is taken from the first occurrence; duplicates contribute only additional page references.

Formatting Results: The terms specified previously might result in the following index:

```

C
conifers 75
    See also pines

P
pines
    Japanese Black 41-44
    See also conifers

W
weasels
    care and feeding 4, 9, 33

```

Note: The example above shows how page devices and PostScript devices treat the index entry headings (C, P, and W). An index formatted for line devices (1403 or IBM 3800 Printing Subsystem Model 1) would have boxes surrounding these headings.

:LIREF—List Item Reference

The :LIREF tag identifies a place in a document where a reference to a list item is made. This tag is useful only for list items for *ordered* lists.

:LIREF	REFID = list-item-id
	[PAGE = YES NO]

Attributes

REFID identifies the list item being referred to. The value must be a unique identifier assigned to a list item with the ID attribute of the :LI tag. The REFID is case sensitive and must duplicate the ID to which it refers.

PAGE indicates whether the page number of the list item be included in the reference. YES means to add the page number even if the list item is on the same page as the reference. NO means do not add the page number even if the list item is on a different page. If the PAGE attribute is not entered, the page number of the list item is included in the reference only if the list item and reference are on different pages.

Usage: References to uniquely identified list items can occur anywhere in text.

Example: The markup for creating an ordered list that includes an ID on a list item is shown in the example in “:SL, :OL, :UL—Lists” on page 78. To refer to that particular list item, you would enter:

```
... See :liref refid=one. for
an example of how to mark up an
ID in a list item.
```

Processing: References can be made to any list items in simple, ordered, and unordered lists where the list-item-ID has been specified. It is the list-item-ID that identifies the item to be referred to when the REFID attribute of the :LIREF tag is specified anywhere in the text.

Specify two or more formatting passes with the FPASSES or TWOPASS option of the SCRIPT command to correctly resolve forward references and to correctly include and resolve the “on page x” phrase. (See “SCRIPT Command Options” on page 117.)

Do not end an input line with this tag. Ending an input line with this tag might cause the next input line to be concatenated to the text inserted by the tag without an intervening blank.

Formatting Results: The example previously shown formats as:

```
... See 1. on page 64 for an example of how to mark up an ID in a list
item.
```

:SL, :OL, :UL—Lists

The :SL, :OL, and :UL tags identify simple, ordered, and unordered lists of items, respectively. The :ESL, :EOL, and :EUL tags identify the ends of the lists. The items of the list are identified by the :LI tag.

:SL, :OL, :UL	[COMPACT]
:ESL, :EOL, :EUL	

:LI—List Item

The :LI tag identifies an item of a list and is valid only within a list.

:LI	ID = list-item-id
-----	-------------------

:LP—List Part

The :LP tag identifies a comment or explanation that applies to a part of a list. The :LP tag allows explanatory material to be inserted into the items that are included in a list. Its use is included in the example below. It is valid within glossary lists, simple lists, ordered lists, unordered lists, and definition lists and should be entered immediately preceding the list items to which it applies.

:LP	
-----	--

Attributes

COMPACT indicates that list items should not be separated from each other by vertical white space. Items are ordinarily separated from each other by a blank line.

ID gives a unique *name* that can be used when referring to list items in an ordered list. (See “:LIREF—List Item Reference” on page 77.) The ID name can be up to seven numbers and uppercase and lowercase letters; the identifier is case sensitive. That is, the same letter is a different value in lowercase from uppercase. (A is different from a, B is different from b, and so on.) No other list item can have the same id name.

If the ID attribute is not entered, no identifier is assigned to the list item.

Usage: Lists can occur anywhere in text and can be nested within other lists, including definition lists and glossary lists.

Each item in a list is identified by a :LI tag; each subset of a list is identified by a :LP tag. List items and list parts are implied paragraphs and can contain any of the text items listed in Figure 4 on page 12. They can also contain any of the basic document elements listed in Figure 5 on page 13, including other lists.

List items are ended by another list item of the same list level, by a nested list, or by the end of the list.

Example: The second item of the following ordered list contains an unordered list, and the unordered list contains a simple list:

```
:ol.  
:li id=one.Item one.  
:li.Item two, composed of:  
:ul compact.  
:li.Unordered list item.  
:sl compact.  
:li.First simple list item  
:li.Second simple list item  
:esl.  
:li.Unordered list item.  
:eul.  
:li.Item three.  
:lp.The remainder of the list  
follows.  
:li.Next-to-last item.  
:li.Last item.  
:eol.
```

Processing: Lists are formatted as *hanging* indent lists, with the item prefix (number, bullet ...) *undented* into the left margin.

Simple list items have no item prefix. The item prefix for each level of nesting for ordered and unordered lists can be individually specified. See Appendix A, "The DSMPROF4 Profile" on page 139 for details.

Formatting Results: The lists previously shown format as:

1. Item one.
2. Item two, composed of:
 - Unordered list item.
 - First simple list item
 - Second simple list item
 - Unordered list item.
3. Item three.

The remainder of the list follows.

4. Next-to-last item.
5. Last item.

:LQ—Long Quotation

The :LQ tag identifies an excerpt quoted from another source, and the :ELQ tag identifies the end of the excerpt. Long quotations are sometimes called *block quotations*.

:LQ	
:ELQ	

Usage: Long quotations can occur anywhere in text and can contain any of the basic document elements listed in Figure 5 on page 13, including other quotations.

Example: The following quotation was taken directly from another document:

```
:p.One of the great speeches
of all time begins like this:
:lq.
:p.Fourscore and seven years ago
our fathers brought forth on
this continent, a new nation, ...
:elq.
:p.It was made by one of the
great presidents of all time.
```

Processing: Long quotations are set off from the text surrounding them by both right and left indentation. They are not surrounded by quotation marks (" "), as are quotations identified by the :Q tag.

Formatting Results: The example shown previously formats as:

One of the great speeches of all time begins like this:

Fourscore and seven years ago our fathers brought forth on this continent, a new nation, ...

It was made by one of the great presidents of all time.

:NOTE—Note

The :NOTE tag identifies a paragraph containing a comment or explanation of particular importance.

:NOTE	
-------	--

Usage: Notes can occur anywhere in text. They are implied paragraphs and can contain any of the text items listed in Figure 4 on page 12.

A note is implicitly ended by a paragraph or a higher-level element.

Example: Here is an example of markup for the :NOTE tag:

```
:note. Notes are simple to use  
within your document.  
Just enter the NOTE tag and follow it  
with the text of the note.
```

Processing: The word “Note” (followed by a colon) is inserted in to the document as a level-two highlighted phrase. The implied paragraph is formatted as a block, without any indentation on the first line.

Formatting Results: The example shown previously formats like this:

Note: Notes are simple to use within your document. Just enter the NOTE tag and follow it with the text of the note.

:P and :PC—Paragraph

The `:P` tag identifies a paragraph—one or more sentences related by their subject matter. The `:PC` tag identifies the continuation of a paragraph that has been interrupted.

<code>:P, :PC</code>	
-----------------------------	--

Usage: Paragraphs can occur anywhere in text and can contain any of the text items listed in Figure 4 on page 12.

A paragraph continuation usually occurs after a figure, example, or long quotation; it indicates that the following text is not a new paragraph, but a continuation of one interrupted by another element.

A paragraph is ended by another paragraph or higher-level element.

Example: The preceding two paragraphs are marked up as:

```
...
:p.
A paragraph continuation usually
occurs after a figure, example, or
long quotation; ...
:p.A paragraph is ended by
another paragraph or
higher-level element.
...
```

A continuation paragraph is marked up as follows to show how a paragraph continues after an unordered list:

```
:p.
Some of our favorite fruits are
also some of the most nutritious:
:ul compact.
:li.Apples
:li.Oranges
:li.Pears
:li.Plums
:eul.
:pc.
This is a good reason to enjoy
them often.
```

This results in the following formatting:

Some of our favorite fruits are also some of the most nutritious:

- Apples
- Oranges
- Pears
- Plums

This is a good reason to enjoy them often.

Processing: The first paragraph after each major heading (level zero through four) is formatted flush left, without any indentation applied to the first line of the paragraph. The second and subsequent paragraphs after each heading are separated from each other by white space, and the first line of each paragraph can be indented.

Paragraph continuations, however, are never indented.

Paragraphs following heading levels five and six are formatted inline with the heading and separated from it by a colon (:).

Formatting Results: The use of the :P tag causes a space to be inserted between lines of text in a document. Each of the paragraphs without heading tags on this page was created using the :P tag.

:*PREFACE*—Preface

The :PREFACE tag identifies introductory remarks, such as acknowledgments, that precede the body of a document.

:PREFACE	
-----------------	--

Usage: A preface appears within the front matter of the document, after the title page. It can contain any of the basic document elements listed in Figure 5 on page 13, as well as level-two through level-six heading segments.

A preface is implicitly ended by a table of contents, figure list, or body element.

Example: The front matter of this document contains the following markup:

```

:
:frontm.
:titlep.

:
:etitlep.
:preface.
:p.The Generalized Markup Language
(GML) is a ...

:
:body.

:
```

Processing: The :PREFACE tag begins a new page. If duplexing is requested with the SYSVAR D option, the new page will be odd-numbered. (See "SYSVAR Option" on page 133.) A level-one heading, *PREFACE*, is placed on the new page.

Note: All the material that appears in the front matter is in single-column format. The front-matter headings do not appear in the table of contents, nor are they numbered. Automatic head-level numbering is suppressed for headings in the front matter, even if requested with the SYSVAR H option. (See "SYSVAR Option" on page 133.)

Formatting Results: The preface will appear at the front of the document, after the title page.

:PSC—Process-Specific Control

The :PSC tag identifies markup and text that are to be processed only under specific conditions, and the :EPSC tag identifies the end of process-specific controls.

:PSC	[proc = 'list of process names']
:EPSC	

Attributes

PROC gives a list of *process names* to which the element applies. If you specify logical devices as the value of PROC, the information within the :PSC and :EPSC tags is ignored if none of the logical devices named with the PROC attribute is valid for the document when it is formatted. If more than one process name is entered, separate the names with blanks and enclose the list in single quotation marks (').

Usage: Process-specific controls can appear anywhere in a document, and can contain any valid tags, text, and control words. They can be used to alter formatting for different output devices, to produce formatting results not provided by the GML starter set, or to fine-tune a document for publication.

Example: If you have the following PSC tag sequence in your document:

```
:psc proc='3800n8'.  
.cm patch for bad page break-  
.cm remove when document is updated  
.pa  
:epsc.
```

the control words between PSC and its end tag are processed only if "3800n8" is specified as the logical device on the SCRIPT command when you request formatting for your document.

During final tuning of a document for publication, undesirable page and column breaks can be removed without permanently affecting the formatting of the document by placing tuning controls within a process-specific control element:

```
:psc proc='tune'.  
.kp 2i  
:epsc.
```

The tuning control word (in this case .kp 2i) is ignored unless the document is formatted for the "tune" process with the SYSVAR P option on the SCRIPT command, like this:

```
sysvar (p tune)
```

(See "SYSVAR Option" on page 133.)

If the PROC attribute is not specified, the element is unconditionally processed. For example, if you specify

```
:psc.  
.sp 4i  
:epsc.
```

the control words between the :psc and :epsc are recognized for all devices on all runs.

Processing: The names of the logical and physical devices for which the document is being formatted (see Table 7 on page 128, Table 9 on page 129, and Table 8 on page 129) are valid process names. Other process names can be specified with the SYSVAR P option (see "SYSVAR Option" on page 133).

Formatting Results: In the "patch for bad page break" example, if you specify "3800N8" as the logical device on the SCRIPT command used to process this document, the text following the :PSC and :EPSC tags begins at the top of the next page of the document. If you *do not* specify "3800N8," the control words within the tags are ignored.

:Q—Quoted Phrase

The :Q tag identifies a phrase in which the exact words of an individual or document are cited, and the :EQ tag identifies the end of the quoted phrase.

:Q	
:EQ	

Usage: Quotations can occur anywhere in text and can contain any of the text items listed in Figure 4 on page 12, including other quotations.

Example:

Lincoln said :q.As I would not
be a slave, so I would not be
a master.:eq.

and

:q.The observation :q.The
coldest winter I ever experienced
was one summer
in San Francisco:eq. is widely
attributed to
Samuel Clemens.:eq.

Processing: Quoted phrases are surrounded with double quotation marks (“ ”); nested quotations alternate between single and double quotation marks.

Formatting Results: The examples previously shown format as:

Lincoln said “As I would not be a slave, so I would not be a master.”

and

“The observation ‘The coldest winter I ever experienced was one
summer in San Francisco’ is widely attributed to Samuel Clemens.”

:RDEF—Row Definition

The :RDEF tag defines the characteristics of a row. This tag allows you to specify cell characteristics such as alignment, arrangement, width, and highlight level; and to name the row definition for future reference by the :TABLE, :ROW, :THD, and :TFT tags. Each row definition used in a table must be defined with an :RDEF tag before the table is started. A row definition cannot be redefined while a table is active.

:RDEF	<p>ID = rdef-id</p> $\left[\text{HP} = \frac{0}{1 \atop 2 \atop 3} \quad [\dots] \right]$ $\left[\text{ALIGN} = \begin{array}{c} \text{LEFT} \\ \text{RIGHT} \\ \text{CENTER} \\ \text{INSIDE} \\ \text{OUTSIDE} \\ \text{JUSTIFY} \end{array} \quad [\dots] \right]$ $\left[\text{CONCAT} = \frac{\text{YES}}{\text{NO}} \quad [\dots] \right]$ $\left[\text{VALIGN} = \begin{array}{c} \text{TOP} \\ \text{BOTTOM} \\ \text{CENTER} \end{array} \quad [\dots] \right]$ $\left[\text{ROTATE} = \begin{array}{c} 0 \\ 90 \\ 180 \\ 270 \\ -90 \\ -180 \\ -270 \end{array} \quad [\dots] \right]$ $\left[\text{MINDEPTH} = \frac{*}{v} \right]$ $\left[\text{CWIDTHS} = \frac{h}{n*} \quad [\dots] \right]$ <p>[ARRANGE = n1 [n2 [n3]].../...]</p> $\left[\text{SHADE} = \begin{array}{c} 0 \\ \text{XLIGHT} \\ \text{LIGHT} \\ \text{MEDIUM} \\ \text{DARK} \\ \text{XDARK} \end{array} \quad [\dots] \right]$
-------	--

Attributes

ID

gives a unique identifier that can be used when referring to this row definition with the :TABLE, :ROW, :THD, or :TFT tag. This attribute is required.

The ID attribute can have up to seven numbers and uppercase and lowercase letters. The identifier is not case sensitive. The identifier "ralph" is the same identifier as "RALPH" or "RaLpH."

HP

specifies the type of highlighting you want for the contents of the cells in the row. The values can be 0, 1, 2, or 3. These values correspond directly to the :HPn tags.

For more information on highlighting, see "Highlighted Phrases" on page 19. This attribute is optional; the default is 0. The highlight level for the cell overrides any highlight level currently in effect.

ALIGN

specifies the horizontal alignment of the contents of the cells. This attribute is optional; the default is LEFT.

You can specify any of six choices for horizontal alignment:

LEFT Aligns the contents of the cell to the left-hand side of the cell.

RIGHT Aligns the contents of the cell to the right-hand side of the cell.

CENTER Aligns the contents of the cell in the middle of the cell.

INSIDE Aligns the contents of the cell toward the inside of the cell. On odd-numbered pages, this aligns the contents of the cell toward the left-hand side of the cell; on even-numbered pages, this aligns the contents of the cell toward the right-hand side of the cell.

OUTSIDE Aligns the contents of the cell toward the outside of the cell. On odd-numbered pages, this aligns the contents of the cell toward the right-hand side of the cell; on even-numbered pages, this aligns the contents of the cell toward the left-hand side of the cell.

JUSTIFY Inserts extra horizontal white space between words in the cell to produce an even left and right edge.

CONCAT

determines whether the input lines are to be concatenated when producing output lines. This attribute is optional; the default is YES.

YES Specifies that the input lines are to be concatenated to produce output lines that are as full as possible.

NO Specifies that the printing of the output lines will match the way they were entered.

VALIGN determines the vertical alignment of the contents of the cells. This attribute is optional; the default is TOP.

You have three choices for vertical alignment:

TOP Indicates that the contents of the cell are to be placed at the top of the cell.

BOTTOM Indicates that the contents of the cell are to be placed at the bottom of the cell.

CENTER Indicates that the contents of the cell are to be centered vertically in the cell.

ROTATE specifies the degree of rotation of the contents of the cell. The allowable values are 0, 90, 180, 270, -90, -180, and -270. This attribute is optional; the default is 0.

This attribute is valid only for AFP page printers and PostScript devices. A composite rotation of 180° or -180° is not allowed for the IBM 3800 Printing Subsystem Model 3 or Model 6.

When formatting for the IBM 3800 Printing Subsystem Model 3 and Model 6, if you specify a composite rotation of 270°, you must ensure that a 270° rotation is available in the default font. The GML required fonts are not shipped with this rotation.

If you specify a rotation of 90, -270, 270, or -90 for a particular cell, the MINDEPTH attribute is required for that cell. See the description of MINDEPTH for more information.

Note: This same attribute is used on the TABLE tag to specify the degree of rotation of the entire table.

MINDEPTH specifies the minimum depth of a cell. Use any of the standard notations for horizontal space units. See Table 1 on page 22 for the list of valid space-unit notations. If the contents of a cell do not fill the minimum depth specified, any white space is added at the top or bottom of the cell (depending on the VALIGN value for the cell) to cause the cell to be as deep as the minimum depth specified. This attribute is optional for cells rotated 0°, 180°, or -180°. If this attribute is not specified for a particular cell, or if an asterisk (*) is specified, the contents of the cell and the depth of other cells in the row determine the depth of the cell.

If you specify a rotation value of 90, -270, 270, or -90 for a particular cell, then the MINDEPTH attribute is required for that cell. In such cells, the MINDEPTH value is used as the column line length for the contents of the cell. The actual depth of such a cell might be greater than the MINDEPTH value specified for that cell, depending on the depth of other cells in the row. However, the contents of the cell are still formatted using the MINDEPTH value. If this produces undesirable formatting results, increase the MINDEPTH value for the cell.

CWIDTHS is used to determine the widths of the individual cells. Use any of the standard notations for horizontal space units. See Table 1 on page 22 for the list of valid space-unit notations.

If the ARRANGE attribute is not used, the CWIDTHS attribute determines the number of cells in the row and the width

of each cell. There will be one cell for each value specified on the CWIDTHS attribute. The cells will be numbered sequentially, starting with the number 1.

If the ARRANGE attribute is used, the number of values specified on the CWIDTHS attribute must be the same as the number of values given between slashes, or if slashes are not used, the number of values on one ARRANGE attribute. The CWIDTHS values indicate the widths of the individual rectangles that were defined with the ARRANGE attribute. The width of a particular cell will then be the sum of the CWIDTHS values for all the grid rectangles that make up the particular cell. Refer to *Document Composition Facility: Generalized Markup Language Starter Set User's Guide* for a step-by-step method for determining the ARRANGE and CWIDTHS values for a cell arrangement.

The asterisk (*) can also be used as an attribute value. This notation indicates that any remaining horizontal space is to be divided equally among the number of asterisks you enter on that particular attribute. A number can precede the asterisk to indicate a factor to apply. For example, a cell with a width of "2*" will be twice as wide as a cell with a width of "*."

If the CWIDTHS attribute is not used, the default is *n* cells of equal width, where *n* is the number of values given between one set of slashes on the ARRANGE attribute or on a single ARRANGE attribute if slashes are not used.

ARRANGE

specifies the arrangement of the cells in a row. You must enter positive integers as values on this attribute. The values you enter correspond to the cell numbers used with the :C tag. This attribute is optional; the default is *n* cells, where *n* is the number of values given on the CWIDTHS attribute. If no CWIDTHS attribute is specified, the default is one cell. Refer to *Document Composition Facility: Generalized Markup Language Starter Set User's Guide* for a step-by-step method for determining the ARRANGE and CWIDTHS values for a cell arrangement. In your input file you can specify the ARRANGE attribute in either of two ways. In the first, use a separate input line for each line of the cell arrangement, like this:

```
ARRANGE='1 1 5 6 7'
ARRANGE='2 3 5 8 8'
ARRANGE='4 4 5 8 8'
```

In the second, specify the arrange attribute on one input line, separating each line of the cell arrangement with a slash and a blank, like this:

```
ARRANGE='1 1 5 6 7 / 2 3 5 8 8 / 4 4 5 8 8'
```

Both examples produce the same results.

SHADE

specifies the type of shading you want for the contents of the cells in the row. This attribute is optional, and the default is 0 for no shading. You have five choices for shading:

XLIGHT
LIGHT
MEDIUM
DARK
XDARK

The SHADE attribute is valid for AFP and PostScript devices only. For AFP devices, the STANDARD shade pattern is used.

Refer to the *Document Composition Facility: SCRIPT/VS Language Reference* for more information on shading.

Usage: The :RDEF tag must be used to create all row definitions you intend to use in a table, before the table is started with the :TABLE tag. The identifier specified on the ID attribute is used on the REFID attribute of the :TABLE, :ROW, :THD, and :TFT tags to indicate that those items are to use this particular row definition.

For the HP, ALIGN, CONCAT, VALIGN, ROTATE, MINDEPTH, and SHADE attributes, more than one value can be specified. The first value applies to the first cell, the second value to the second cell, and so on. If fewer values are specified than the number of cells, the remaining cells use the last value specified. If more values are specified than the number of cells, the extra values are ignored. If multiple values are given, the entire string of values must be enclosed in single quotation marks. The values for all of the attributes, except for ROTATE and MINDEPTH, can be given as a minimum unique abbreviation. For example, on the ALIGN attribute, the values "LEFT," "LEF," "LE," and "L" are all valid ways to specify left alignment for the contents of the cell. The ROTATE and MINDEPTH attributes use numbers and vertical space-unit notations as values, so these values cannot be abbreviated.

Example: See "Examples of Tables" on page 99 for examples of the use of the :RDEF tag in conjunction with the :TABLE, :ROW, and :C tags.

Processing: When the :RDEF tag is processed, GML macro DSMRDEF generates a table definition control word (.TD) to process the ARRANGE attributes (if specified). Although the ARRANGE definition might have been defined using a separate input line for each line of the cell arrangement, the entire ARRANGE definition for a given row is processed as one input line. The parameter list is then placed into a symbol array and used as the ARRANGE parameter list. If the entire table definition (.TD), including control word, attributes, and parameters (after symbol substitution) exceeds 256 characters, truncation occurs and unpredictable errors result.

:TABLE—Table

The :TABLE tag indicates the start of a table. After starting the table, use the :ROW and :C tags to start the various rows and cells of the table. The :ETABLE tag identifies the end of a table.

:TABLE	REFID = rdef-id [ID = table-id] [SPLIT = $\frac{\text{NO}}{\text{YES}}$] $\left[\begin{array}{l} \text{ROTATE} = \frac{0}{90} \\ 180 \\ 270 \end{array} \right]$ $\left[\begin{array}{l} \text{PAGE} \\ \text{COLUMN} \end{array} \right]$ [WIDTH = h]
:ETABLE	

Attributes

REFID identifies the row definition being referred to. The value must be a unique identifier assigned to a row definition with the ID attribute on the :RDEF tag. This attribute is required. The identifier is not case sensitive.

ID specifies a unique identifier that can be used when referring to the table with the :TREF tag. The identifier can be up to seven numbers and uppercase and lowercase letters; the identifier is case sensitive. No other table can have the same identifier.

SPLIT specifies whether or not a table can be split between rows when the table is too large to fit in a single column. This attribute is optional; the default is NO.

NO Specifies that if the table does not fit in the current column, it cannot be split except prior to rows that have SPLIT = YES specified on the :ROW tag.

YES Specifies that if the table does not fit into the current column, it can be split prior to any row, except for rows that have specified SPLIT = NO on the :ROW tag.

You can also specify the SPLIT attribute on the :ROW tag. See the description of the :ROW tag for more information.

ROTATE specifies a rotation for the entire table. The allowable values are 0, 90, -270, 180, -180, 270, and -90. This attribute is optional; the default is 0. This attribute is valid only for AFP page printers and PostScript devices.

The fonts needed to print rotated text might not be available at your installation. For example, when formatting for the IBM 3800 Printing Subsystem Model 3 and Model 6, if you specify a

composite rotation of 270°, you must ensure that a 270° rotation is available.

A table will not be rotated if the COLUMN attribute is specified. If the ROTATE attribute is specified, the table caption and the table description, if used, are not rotated.

A composite rotation of 180° or -180° is not allowed for the IBM 3800 Printing Subsystem Model 3 and Model 6.

PAGE sets the width of the table to the current line length. The COLUMN and PAGE attributes are mutually exclusive. The last one specified is used. PAGE is the default.

COLUMN sets the width of the table to the current column line length. The COLUMN and PAGE attributes are mutually exclusive. The last one specified is used.

WIDTH specifies the width of the table. The value specified can be any of the horizontal space units described in Table 1 on page 22. If this attribute is not specified, the width of the table is determined by the COLUMN or PAGE attribute.

If this attribute is specified with the COLUMN or PAGE attribute, the WIDTH attribute value is used.

Usage: The :TABLE tag is used to start a table. The REFID attribute is required and must refer to a row definition previously defined with the :RDEF tag. The row definition specified determines the layout of the rows in the table, unless another row definition is specified on the REFID attribute of the :ROW tag when starting a row. While a table is active, the :ROW and :C tags indicate the start of rows and cells.

Example: See "Examples of Tables" on page 99 for examples using the :TABLE tag.

Processing: If SPLIT=YES has been specified on the :TABLE tag, and if the table does not fit in the current column, an attempt is made to split the table before any row that has not specified SPLIT=NO on the :ROW tag. The split that puts as much of the table in the current column as possible is used.

If SPLIT=NO has been specified on the :TABLE tag, the table can be split only before rows that specify SPLIT=YES on the :ROW tag. If there are no such rows, the entire table is moved to the next column. If the table does not fit in an empty column, a message is issued, and the table is split, placing as many rows in the current column as possible. If the table is split, a table footer is not placed at the end of the table in the current column. The rest of the table is placed starting in the next column, but the table header is not placed in the next column.

:ROW—Row

The :ROW tag indicates the start of a row in a table and is valid only while a table is active. After the row has been started, use the :C tag to start the cells in the row.

:ROW	[REFID = rdef-id] [SPLIT = $\frac{\text{NO}}{\text{YES}}$]
:EROW	

Attributes

REFID identifies the row definition being referred to. The value must be a unique identifier assigned to a row definition with the ID attribute on the :RDEF tag. The identifier is not case sensitive.

SPLIT specifies whether or not the table can be split just ahead of this row. This attribute is optional; if not specified, the SPLIT value specified on the :TABLE tag is used.

NO Specifies that if the table does not fit in the current column, it cannot be split prior to this row.

YES Specifies that if the table does not fit in the current column, it can be split prior to this row.

Usage: The :ROW tag indicates the start of a row. The row definition referred to by the REFID attribute is used in formatting the cells within the row.

If the REFID attribute is not used, the row definition from the previous row is used. If this is the first row in the table, the row definition specified on the REFID attribute on the :TABLE tag is used.

After a row has been started, the :C tag is used to indicate the start of the individual cells within the row.

The :EROW tag ends a row, but this tag is optional. A :TNOTE, :TCAP, :TDESC, :ETABLE, or another :ROW tag also ends a row.

If the SPLIT attribute is not specified, the SPLIT value used on the :TABLE tag determines whether or not the table can be split prior to this row.

Example: See "Examples of Tables" on page 99 for examples using the :ROW tag.

:C—Cell

The :C tag is used to create each compartment or cell in the rows of a table. It is valid only while a row, table header, or table footer is active. The text to be placed in a cell follows the :C tag.

:C	[n]
:EC	

Attributes

n is the number of the cell to be started. This attribute is optional.

Usage: The :C tag is used to start a cell after a :ROW, :TFT, or :THD tag has been specified. The row definition specified on the :ROW, :TFT, or :THD tag determines the highlight level used for text, horizontal alignment, vertical alignment, cell width, rotation, and minimum depth of the cell.

The cell contents can continue for more than one input line. A cell can be ended by the :EC tag, another :C tag, a :ROW tag, or the :ETABLE tag. A :TNOTE, :ETHD, :ETFT, :TCAP, or :TDESC tag also ends a cell. Any basic document element or lines of text can occur within a cell except figures, examples, footnotes, or other tables.

If you place :C tags on the same line with blanks between them, you will get unexpected results because the blanks are formatted as text in the cell.

Example: See "Examples of Tables" on page 99 for examples on using the :C tag.

Processing: If a cell number is specified for a cell that does not exist, an error results. If a cell number is not given, the next higher cell number in the row is started. If no such cell exists, the current row is ended and a new row is started, using the same row definition, and the cell with the lowest cell number in that row is started. If the first occurrence of the :C tag does not use a cell number, the cell with the lowest cell number is started.

When the same cell number is specified on more than one :C tag within a row, that cell will be repeated by terminating the earlier occurrence of the cell with a horizontal rule and restarting the cell with the additional text.

If the NO value on the CONCAT attribute is specified for a cell, the normal formatting of text (justification and concatenation) is suspended within the cell; each input line becomes a separate output line. Other GML tags should be used with caution within the cell because concatenation of input lines is suppressed. Unexpected line breaks or concatenation of source text lines can occur when GML tags are used within cells in this case.

Note: Processing might be incorrect when text is intermixed with GML tags inside of cells, examples, and figures.

:TFT—Table Footer

The :TFT tag is used to define a table footer. A table footer is placed at the bottom of the table, and at the bottom of all subsequent columns or pages where the table appears.

:TFT	[REFID = rdef-id]
:ETFT	

Attributes

REFID identifies the row definition being referred to. The value must be a unique identifier assigned to a row definition with the ID attribute on the :RDEF tag. The identifier is not case sensitive.

Usage: The :TFT tag is not recognized outside the scope of a table. The table footer must be defined before the first row in a table is started. The :ETFT tag must be used to end the table footer definition, and it must appear starting in column one of an input line.

If the REFID attribute is not specified, the row definition specified on the :TABLE tag or the :THD tag (if a table header precedes the :TFT tag in this table) is used to determine the row definition to use for the table footer.

Use the :C tag within your table footer definition to fill the different cells in your table footer. The :ROW tag is not allowed in a table footer.

Specify SPLIT = YES on the :RDEF tag if the table is to be on multiple pages. This ensures that the table footer appears on subsequent pages.

:THD—Table Header

The :THD tag is used to define a table header. The table header is placed at the top of the table, and at the top of all subsequent columns or pages where the table appears.

:THD	[REFID = rdef-id]
:ETHD	

Attributes

REFID identifies the row definition being referred to. The value must be a unique identifier assigned to a row definition with the ID attribute on the :RDEF tag. The identifier is not case sensitive.

Usage: The :THD tag is not recognized outside the scope of a table. The table header must be defined before the first row in a table is started. The :ETHD tag must be used to end the table header definition, and it must appear starting in column one of an input line.

If the REFID attribute is not specified, the row definition specified on the :TABLE tag or the :TFT tag (if a table footer precedes the :THD tag in this table) is used to determine the row definition to use for the table header.

Use the :C tag within your table header definition to fill the different cells in your table header. The :ROW tag is not allowed in a table header.

Specify SPLIT = YES on the :RDEF tag if the table is to be on multiple pages. This ensures that the table header appears on subsequent pages.

Example: See "Examples of Tables" on page 99 for examples using the :THD tag.

:TCAP—Table Caption

The :TCAP tag identifies a short title or **caption** for the table and is valid only within a table. A caption for a table is necessary if the table is to be numbered for cross-referencing and for inclusion in the list of tables.

:TCAP	.table caption
-------	----------------

Usage: The text for the table caption must be entered all on the same line as the TCAP tag, *or* it can be entered entirely on the line following the tag. The table caption cannot contain any other GML tags.

Tables must be captioned to resolve references correctly.

If the table is to be rotated using the ROTATE attribute of the :TABLE tag, the table caption and description will not be rotated.

Use the TCAP tag at the end of your table, prior to the TABLE end tag.

Example: See "Examples of Tables" on page 99 for examples using the :TCAP tag.

Processing: The table caption is placed beneath the table, followed by any table description text.

If a table caption is specified, the table is assigned a number. This number is included in the formatted table caption. The table number and caption are included in the List of Tables that can be placed in the front matter of the document by using the :TLIST tag. If no caption is specified, the table is not numbered and the caption cannot, therefore, appear in the List of Tables.

:TDESC—Table Description

The :TDESC tag identifies text describing the table and is valid only within a table.

:TDESC	
--------	--

Usage: Any further descriptive text following the table caption requires the :TDESC tag.

The table description is started by the :TDESC tag. Any basic document element or lines of text can occur within the table description except figures, examples, footnotes, or other tables. The description can continue for more than one input line and is ended by the :ETABLE tag.

If the table is to be rotated using the ROTATE attribute of the :TABLE tag, the table caption and description will not be rotated.

Example: See "Examples of Tables" on page 99 for examples using the :TDESC tag.

Processing: The table description is placed beneath the table, immediately following the table caption, if any.

Examples of Tables

The following examples use the different table tags.

One-Row, Column-Wide Table

A row in which all cells extend from the top of the row to the bottom of the row does not require the use of the `ARRANGE` attribute on the `:RDEF` tag. The `CWIDTHS` attribute determines the width of the cells, and the cells are numbered sequentially, from left to right, starting with 1.

The following shows the input for a table that consists of a single row with four cells. The first cell is two inches wide; the other three cells are equal in width and fill up the rest of the table. The `COLUMN` attribute is used on the `:TABLE` tag to cause this table to be as wide as the current column. The `:TCAP` and `:TDESC` tags are used to create a table caption and description for the table.

```
:rdef id=examp11 cwidths='2i * * *'.
:table id=table3 refid=examp11 column.
:row.
:c.This is the text that goes into the first cell.
This is the widest cell in this row.
:c.This is the text in the second cell.
This cell will have more text in it than the third
and fourth cells.
:c.This is in the third cell.
:c.This is text that goes into the fourth cell.
:tcap.Example of a Column-Wide Simple Table
:tdesc.A column-wide simple table with four cells.
:etable.
```

Note: In this example we use the `ID` attribute of the `:TABLE` tag because we refer to this table in our discussion of the `:TREF` tag.

Formatting Results

The above example produces the following results:

This is the text that goes into the first cell. This is the widest cell in this row.	This is the text in the second cell. This cell will have more text in it than the third and fourth cells.	This is in the third cell.	This is text that goes in the fourth cell.
--	---	----------------------------	--

Table 3. Example of a Column-Wide Simple Table. A column-wide simple table with four cells.

Note: If this book had been formatted with the starter set, a colon would appear after the table caption instead of a period.

One-Row, Page-Wide Table

Let's repeat the above example, but this time the COLUMN attribute will not be specified on the :TABLE tag. Therefore, the default of PAGE will be used, and the table will be as wide as the current line length.

```
:rdef id=examp12 cwidths='2i * * *'.
:table refid=examp12.
:row.
:c.This is the text that goes into the first cell.
This is the widest cell in this row.
:c.This is the text in the second cell.
This cell will have more text in it than the third
and fourth cells.
:c.This is in the third cell.
:c.This is text that goes into the fourth cell.
:tcap.Example of a Page-Wide Simple Table
:tdesc.A simple table with four cells.
:etable.
```

Formatting Results

The above example produces the following results:

This is the text that goes into the first cell. This is the widest cell in this row.	This is the text in the second cell. This cell will have more text in it than the third and fourth cells.	This is in the third cell.	This is text that goes into the fourth cell.
--	---	----------------------------	--

Table 4. Example of a Page-Wide Simple Table. A simple table with four cells.

Note: If this book had been formatted with the starter set, a colon would appear after the table caption instead of a period.

Table Using the ARRANGE Attribute

A row that contains at least one cell that does not extend from the top of the row to the bottom of the row requires the use of the ARRANGE attribute to specify the arrangement of the cells. Using the ARRANGE attribute can be a bit tricky. The description of tables in the *Document Composition Facility: Generalized Markup Language Starter Set User's Guide* contains a step-by-step method for determining the ARRANGE and CWIDTHS attributes to produce a desired cell arrangement.

The following table uses the ARRANGE attribute. Also notice that a factor was used in determining the width of the first cell on the CWIDTHS attribute. The ALIGN and VALIGN attributes are also used to specify horizontal and vertical alignments other than the default of LEFT (for horizontal alignment) and TOP (for vertical alignment).

```

:rdef id=thead cwidths='2* * * * *'
      arrange='1 2 3 3 3 3'
      arrange='1 2 4 5 6 6'
      arrange='1 2 4 5 7 8'
      align='left left center center center left center'
      valign='center center top center center top bottom'.
:table refid=thead.
:row.
:c.Selected major industry and selected country
:c.Number of U.S. corporation returns
:c.Controlled foreign corporations
:c.Number of foreign corporations
:c.Total assets
:c.Foreign corporations with current earnings and
profits (+) before taxes
:c.Current earnings and profits before taxes
:c.Foreign income taxes (net)
:tcap.A More Complex Table
:tdesc.This table requires the use of the ARRANGE attribute to
define the desired arrangement of cells.
:etable.

```

Formatting Results

The above example produces the following results:

Selected major industry and selected country	Number of U.S. corpo- ration returns	Controlled foreign corporations			
		Number of foreign corpo- rations	Total assets	Foreign corporations with current earnings and profits (+) before taxes	
				Current earnings and profits before taxes	Foreign income taxes (net)

Table 5. A More Complex Table. This table requires the use of the ARRANGE attribute to define the desired arrangement of cells.

Note: If this book had been formatted with the starter set, a colon would appear after the table caption instead of a period.

Multiple-Row Table

A table can be constructed by combining several rows. Different row definitions can be used. The following table uses three row definitions:

```
:rdef id=row1 cwidths='1i * * 3*'.
:rdef id=row2 cwidths='* *'.
:rdef id=row3.
:table column refid=row1.
:row.
:c.first cell:c 4.fourth cell:c 3.third cell:c 2.second cell
:row refid=row2.
:c.Only two cells in this row.:c.This is the other cell.
:row refid=row3.
:c.This row contains only one cell.
:etable.
```

Note: We intentionally filled the cells out of sequence to show you that they format correctly.

Formatting Results

The preceding example produces the following results:

first cell	second cell	third cell	fourth cell
Only two cells in this row.			This is the other cell.
This row contains only one cell.			

Table with Repeated Cells

A cell can be repeated within a row. When the same cell number is specified on more than one :C tag within a row, that cell will be repeated by terminating the earlier occurrence of the cell with a horizontal rule. The cell is restarted with the additional text.

```
:rdef id=gmltags arrange='1 2' cwidths='* 2*'.
:table refid=gmltags.
:row.
:c.definition list (:DL)
:c 2.definition term heading (:DTHD)
:c 2.definition description heading (:DDHD)
:c 2.definition term (:DT)
:c 2.definition description (:DD)
:c 2.list part (:LP)
:row.
:c.figure (:FIG)
:c 2.figure caption (:FIGCAP)
:c 2.figure description (:FIGDESC)
:etable.
```

Formatting Results

The preceding example produces the following results:

definition list (:DL)	definition term heading (:DTHD)
	definition description heading (:DDHD)
	definition term (:DT)
	definition description (:DD)
	list part (:LP)
figure (:FIG)	figure caption (:FIGCAP)
	figure description (:FIGDESC)

Table with Cells with Intervening Blanks

If you place :C tags on the same line with blanks between them, you will get unexpected results because the blanks are formatted as text in the cell. The following example shows how the text in cells formats when you enter the :C tags on the same line with blanks between them and specify ALIGN= CENTER.

```
:rdef id=cexp cwidths='1i 1i 1i 1i'
      align=center.
:table refid=cexp.
:thd.
:c.Heading1
:c.Heading2
:c.Heading3
:c.Heading4
:ethd.
:row.
:c.cell1      :c.cell2      :c.cell3      :c.cell4
:row.
:c.cell5
:c.cell6
:c.cell7
:c.cell8
:etable.
```

Formatting Results

The preceding example produces the following results:

Heading1	Heading2	Heading3	Heading4
cell1	cell2	cell3	cell4
cell5	cell6	cell7	cell8

The text in the cells entered with blanks does not appear to align in the center of the cell as it should when specifying ALIGN= CENTER because the blanks are formatted as if they were additional text in the cells.

Table with Table Header, Table Caption, and Table Description

The following table uses several of the different attributes on the `:RDEF` tag:

- The `ALIGN` attribute is used to specify different horizontal alignments for different cells.
- The `VALIGN` attribute is used to specify different vertical alignments of the cells.
- The `ROTATE` attribute is used to rotate some of the cells that describe the information contained in the table; for these cells, the `MINDEPTH` attribute is also specified.
- The `ARRANGE` attribute is necessary for the row definition named "xmp5," as some of the cells do not extend from the top of the row to the bottom of the row.
- The `CWIDTHS` attribute is used to determine the width of the cells; notice that in the row definition named "body" the `CWIDTHS` attribute also determines the number of cells in the row.

Notice that in the definition for the row named "xmp5," the `CWIDTHS` attribute contains only one value, but there are eight cells in the row. The other `CWIDTHS` values will use the default of '*', which means their widths will be equal and will fill out the rest of the row.

A table header has been defined to show the relationship of the data in the different cells in the table. If the table spans more than one page, the table header will appear at the top of each page the table appears on.

This table also contains a table caption and description to provide an explanation of the purpose of this table.

Note: If this book had been formatted with the starter set, a colon would appear after the table caption instead of a period.

```

:rdef id=xmp5 cwidths='10p * * * * *' concat=n hp=2
      arrange='1 2 3 4 4 4 4 / 1 2 3 5 6 7 8 9'
      align='center left left center left' valign=center
      rotate='0 270 270 0 270' mindepth='* 1.6i 1.6i * 1.4i'.
:rdef id=body cwidths='10p * * * * *'
      hp='2 0' align='left center'.
:table refid=xmp5 column.
:thd.
:c.Type of Offense
:c.Number of Offenders
:c.Number of Offenses
:c.Distance from Offender's Residence
:c.Less than 1 Block
:c.1 to 2 Blocks
:c.2 to 3 Blocks
:c.3 to 4 Blocks
:c.Unknown
:ethd.
:row refid=body.
:c.Trespassing:c.02:c.03:c.40:c.20
:row.:c.Carrying Weapons:c.05:c.04:c.29:c.30
:row.:c.Drinking, disorderly conduct,
creating disturbance:c.07:c.10:c.40:c.30
:row.:c.Subtotal:c.14:c.07:c.79:c.50:c.40:c.30
:tcap.Place of Offense and Residence of Offender
:tdesc.This table indicates the correlation between different
types of criminal offenses and the distance from the
offender's residence to the scene of the offense.
:etable.

```

Formatting Results

The above example produces the following results:

Type of Offense	Number of Offenders	Number of Offenses	Distance from Offender's Residence				
			Less than 1 Block	1 to 2 Blocks	2 to 3 Blocks	3 to 4 Blocks	Unknown
Trespassing	02	03	40	20			
Carrying Weapons	05	04	29	30			
Drinking, disorderly conduct, creating disturbance	07		10		40		30
Subtotal	14	07	79	50	40		30

Table 6. Place of Offense and Residence of Offender. This table indicates the correlation between different types of criminal offenses and the distance from the offender's residence to the scene of the offense.

Using the SHADE Attribute

```
:tdef id=shade cwidths='* *' arrange='1 2 / 1 3'.  
:table width=column refid=shade frame=box rules=both  
      shade='xlight light medium'.  
:row.  
:c.First cell in the table  
:c.Second cell in table  
:c.Third cell in table  
:etable.
```

Formatting Results

The above example produces the following results:

First cell in the table	Second cell in the table
	Third cell in the table

:TLIST—List of Tables

The :TLIST tag identifies the place in the document where a list of tables should be created.

:TLIST	
---------------	--

Usage: Normally, the list of tables is placed in the front matter of a document. If the :TLIST tag is not entered, a list of tables is not created.

Example: This document contains:

```
:gdoc.  
:frontm.  
  
:  
:toc.  
:figlist.  
:tlist.  
:body.  
  
:  
:egdoc.
```

Processing: The list of tables is placed on a new page preceded by the heading, "List of Tables." If duplexing has been requested with the SYSVAR D option, the list is placed on an odd-numbered page. (See "SYSVAR Option" on page 133.) Those tables in the document that have table captions, as identified with the :TCAP tag, are listed by table number, along with the table caption and page number.

When the :TLIST tag is placed in the front matter, the list of tables will be empty unless two or more formatting passes are specified using the FPASSES or TWOPASS option of the SCRIPT command. (See "SCRIPT Command Options" on page 117.) The list of tables will be empty because the entries in the list of tables are references to tables defined later in the document. Before final formatting of a document for publication, you may want to place the :TLIST tag temporarily in the back matter so you can run the book with a single pass. If you place your :TLIST in the back matter, it will format in two-column style.

Formatting Results: The front matter of this document contains a table list.

:TNOTE—Table Note

The :TNOTE tag inserts a note into a table. This tag starts a new row, consisting of one cell that is as wide as the table. The :ETNOTE tag ends the table note.

:TNOTE	
:ETNOTE	

Usage: Use the :TNOTE tag to include notes in a row. The table note is placed in a row with one cell that is as wide as the table. Do not use the :ROW or :C tags within a table note. After the :ETNOTE tag has been used to end the table note, you must use a :ROW tag to start another row in the table.

Example: The following shows the input required to create a very simple table that contains a table note:

```
:rdef id=noteex cwidths='* *'.
:table column refid=noteex.
:row.
:c.This is the first cell in this table.
:c.This is the second cell in this table.
:tnote.This is a note that appears
in the middle of the table.:etnote.
:row.
:c.This is in the second row of this table.
:etable.
```

Processing: When the :TNOTE tag is found, the current cell and row are ended. A new row that contains a single cell that is as wide as the table is started. The word “**Note:**” in a level-two highlighted phrase (HP2) is generated and printed at the beginning of any note text entered. When the :ETNOTE tag is found, the cell and the row that contain the table note are ended.

Formatting Results: The above example produces the following results:

This is the first cell in this table.	This is the second cell in this table.
Note: This is a note that appears in the middle of the table.	
This is in the second row of this table.	

:TREF—Table Reference

The :TREF tag identifies a place in a document where a reference to a captioned table should be made.

:TREF	REFID = table-id [PAGE = $\frac{\text{YES}}{\text{NO}}$]
-------	---

Attributes

REFID identifies the table being referred to. The value must be a unique identifier assigned to a table with the ID attribute of the :TABLE tag. The REFID is case sensitive and must duplicate the ID to which it refers. This attribute is required.

PAGE indicates whether the page number of the table be included in the reference. YES means to add the page number even if the table is on the same page as the reference. NO means do not add the page number even if the table is on a different page. If the PAGE attribute is not entered, the page number of the table is included in the reference only if the table and the table reference are on different pages.

Usage: References to uniquely identified tables can occur anywhere in text.

Example: A reference to a table might appear in your text as follows:

The unique identifier
of :tref refid=table1. is table3.

Processing: Tables must be captioned to resolve references correctly. It is the caption that assigns a unique number to a table. Specify two or more formatting passes with the FPASSES or TWOPASS option of the SCRIPT command to correctly resolve forward references and to correctly include and resolve the "on page x" phrase. (See "SCRIPT Command Options" on page 117.)

Even when FPASSES is used, inclusion of the "on page x" phrase might be incorrect if the table reference is placed on the first line of the same page as the table being referenced, or on the first line of the following page. Manual tuning may be necessary to provide correct inclusion of the "on page x" phrase. Use the PAGE attribute of the TREF tag for this purpose.

Do not end an input line with this tag, because that might cause the next input line to be concatenated to the text inserted by the tag, without an intervening blank.

Formatting Results: The unique identifier of Table 3 on page 99 is table3.

:TITLEP—Title Page

The :TITLEP tag identifies a front matter element that contains general information about the document. The :ETITLEP tag identifies the end of the title page.

:TITLEP	
:ETITLEP	

:TITLE—Document Title

The :TITLE tag identifies the title of a document. It can appear only within the title page element.

:TITLE	[STITLE = 'short title']	.document title
---------------	--------------------------	------------------------

Attribute

STITLE identifies a short title for a document. The value can be any character string, but if the short title contains any blanks or special characters, it must be enclosed in single quotation marks ('').

:DOCNUM—Document Number

The :DOCNUM tag identifies a number associated with a document, such as a form, serial, or order number. It can appear only within the title page element.

:DOCNUM		.document number
----------------	--	-------------------------

:DATE—Document Date

The :DATE tag identifies the date of a document. It can appear only within the title page element.

:DATE		.[document date]
--------------	--	-------------------------

:AUTHOR's—Author Name

The :AUTHOR tag identifies the writer of a document. It can appear only within the title page element and can be repeated as often as necessary.

:AUTHOR		.author name
----------------	--	---------------------

Usage: The title page element should be the first element in the front matter. It can contain:

- The :TITLE tag, which can appear anywhere within a title page element and can specify any title.
- The :DOCNUM tag, which can appear anywhere within a title page element and can specify any number. If the :DOCNUM tag is not entered, no number is assumed.
- The :DATE tag, which can appear anywhere within a title page element and indicates that the document is to be dated. Any date can be specified; if no date is specified, the current date is used.
- The :AUTHOR tag, which can appear anywhere within a title page element and can be repeated to identify coauthors.
- The :ADDRESS tag (see “:ADDRESS—Address” on page 38), which can appear anywhere within a title page element and can be repeated as often as necessary.

The document title, document number, date, and author name must each be entered on a single line in the source document and cannot contain any other tags. The author tag can be repeated as necessary if there is more than one author.

This document contains:

```
:titlep.  
:title stitle='GML Starter Set Reference'.  
Generalized Markup Language Starter Set Reference  
:docnum.SH20-9187-05  
:date.  
:author.Sue Smith  
:address.  
:aline.Information Development XIX  
:aline.Silverado, Colorado  
:eaddress.  
:etitlep.
```

Processing: The information identified in the title page element is used to create a title page, as indicated by the SYSVAR T option of the SCRIPT command. (See “SYSVAR Option” on page 133.)

If duplexing is requested with the SYSVAR D option, the document title (or the shortened version given with the STITLE attribute) is placed at the bottom of each page until a major section heading is encountered, such as a table of contents, an abstract, or a :H1 tag. Thereafter, the document title appears at the bottom of only the even-numbered pages.

When duplexing is not in effect, the document title (or shortened title) is placed at the bottom of each page until a major section heading is encountered, such as a table of contents, an abstract, or a :H1 tag. Thereafter, the contents of the level-one heading tag or its short title specified with the STITLE attribute appears at the bottom of each page.

If a document date is given, the SCRIPT/VS symbol &date is replaced with the document date, rather than the current date. (The &date symbol is described in “SCRIPT/VS Symbols” on page 23.)

Formatting Results: Figure 10 illustrates the title page created by the title page element given previously.

GENERALIZED MARKUP LANGUAGE STARTER SET REFERENCE

Document Number SH20-9187-06

April 7, 1991

Sue Smith

Information Development XIX
Boulder, Colorado

Figure 10. The Title Page of a Document. This title page is created from the information provided in the title page element of the front matter.

:TOC—Table of Contents

The :TOC tag identifies a place in the document where a table of contents should be created.

:TOC	
-------------	--

Usage: The table of contents is normally placed in the front matter of a document, between the abstract or preface and the list of illustrations. If the :TOC tag is not entered, a table of contents is not created.

Example: This document contains:

```
:gdoc.  
:frontm.  
  
:  
:preface.  
  
:  
:toc.  
:figlist.  
:body.  
  
:
```

Processing: The table of contents is placed on a new page, preceded by the heading "Table of Contents." If duplexing is requested with the SYSVAR D option, the page will be odd-numbered and the heading will be right justified unless using offset style. (See "SYSVAR Option" on page 133.) Headings of level zero through level four that appear in the document's body, appendixes, and back matter are listed, along with their page numbers. Headings of level zero and level one are highlighted.

When the :TOC tag is placed in the front matter, the table of contents will be empty unless two or more formatting passes are specified with the FPASSES or TWOPASS option of the SCRIPT command. (See "SCRIPT Command Options" on page 117.)

Formatting Results: A table of contents is formatted.

:XMP—Example

The :XMP tag identifies an example or illustration that appears where it is placed, with text surrounding it. Examples differ from figures in that they are not captioned nor are they referred to from other parts of the document. The :EXMP tag identifies the end of an example.

:XMP	[DEPTH = vertical-depth]
:EXMP	

Attributes

DEPTH specifies that an amount of vertical white space be included in the example before the text of the example. The DEPTH attribute value can be given in any of the space-unit notations illustrated in Table 1 on page 22.

If the DEPTH attribute is not entered, the example is as large as necessary to contain the text between the :XMP and :EXMP tags, up to one full column.

Usage: Examples can occur anywhere in text, except within other examples, tables, figures, or footnotes. They can, however, include references to figures, tables, or footnotes, in addition to references to headings and ordered list items. The body of an example is composed of all the text between the :XMP tag and the :EXMP tag. It can contain basic document elements (except other examples, figures, or footnotes), lines of text, character graphics, control words, and macros.

Example: An example is marked up as:

```
Text written before the example
:xmp depth=1i.
...
When this example is formatted,
there will be one inch of white
space (plus the vertical white
space that precedes every example)
before this text appears.
...
:exmp.
Text written after the example.
```

Processing: The normal formatting of text (justification and concatenation) is suspended within an example; each input line becomes a separate output line. Other GML tags should be used with caution within examples because concatenation of input lines is suppressed. Unexpected line breaks or concatenation of source lines may occur when GML tags are used within examples.

The GML Starter Set indents examples two character spaces from the left margin. If spelling verification has been requested for a document, it is suspended inside examples. See the SPELLCHK option of the SCRIPT command (as described in "SCRIPT Command Options" on page 117).

The formatting environment, which includes such things as indentation, centering, and character translation, is saved at the beginning of an example and restored at the end of an example. The active formatting environment is saved and restored around examples. The active formatting environment is described in more detail in the *Document Composition Facility: SCRIPT/VS Text Programmer's Guide*.

Note: Processing might be incorrect when text is intermixed with GML tags inside of cells, examples, and figures.

Formatting Results: The example given previously formats as:

Text written before the example

When this example is formatted,
there will be one inch of white
space (plus the vertical white
space that precedes every example)
before this text appears.

Text written after the example.

Chapter 4. Processing GML Documents with SCRIPT/VS

To process your documents, you need to know how to invoke SCRIPT/VS. You also need to be familiar with some of the SCRIPT command options.

The command that invokes SCRIPT/VS is "SCRIPT"; the name of the document to be processed is entered immediately after the command, separated from the command name by at least one blank:

```
script mydoc
```

The document name is followed by any options to be used. Some of the options of the SCRIPT command that you might want to use are described below. Your installation's procedures may require you to specify options in addition to those you choose from this manual. For more detailed information on the SCRIPT command and its options, refer to the *Document Composition Facility: SCRIPT/VS Language Reference*.

SCRIPT Command Options

The first parameter of the SCRIPT command is an identifier of the file or data set that contains your document. Options follow the document name.

In CMS, options must be separated from the document name by a left parenthesis:

```
script mydoc ( options
```

In TSO (see "TSO Environment" on page 131) and CICS (see "ATMS-III System" on page 132), options are separated from the document name by at least one blank:

```
script mydoc options
```

Options must be separated from each other by at least one blank. Some options have values or *suboptions* that must be enclosed in parentheses.

In the descriptions below, the options are followed by the values enclosed in parentheses. In CMS and TSO, you can enter the entire command in either uppercase or lowercase letters, whichever is more convenient for you. In CICS, the command must be entered in lowercase letters.

Processing Options

BIND(obind ebind):

Specifies that the printed portion of the output page is to be shifted to one side to leave room for binding. It is shifted 'obind' space units to the right on odd-numbered pages, and 'ebind' space units to the right on even-numbered pages. (You would normally make 'ebind' smaller than 'obind' for duplex printing, so even-numbered pages would have a large enough right margin for binding.)

The 'ebind' value can be omitted when you want all pages shifted the same amount (as you normally would for printing on one side of the paper).

If you don't specify any bind, the default depends on the device type (see Table 7 on page 128 for the bind values used for each device type).

CHARS(font1 ... font32):

Specifies the fonts to be used when formatting. If CHARS is not specified, the default font specified for the logical device is used. At least one font must be specified when you use the CHARS option. A maximum of 31 fonts can be specified.

For the 3800, Model 1: You can specify two uppercase and lowercase fonts — the first is the normal text font, and the second is the font used for headings, highlighting, and so forth. For only uppercase fonts, you can specify four. Refer to the *Document Composition Facility: SCRIPT/VS Text Programmer's Guide* for the fonts that are available for the 3800.

Note: In addition to the font specification on the SCRIPT command, a compatible font specification must be given in the job control statements for printing the document on the 3800 Model 1. Consult your supervisor or the document administrator in your organization to ensure the correct font specifications have been made.

For Page Devices and PostScript Devices: You should consult the document administrator or your supervisor to see what fonts are available when you request formatting for page devices or PostScript devices. See "Font Requirements for Page Printers" on page 126 and "Overriding the Default Font Using the CHARS Option" on page 127 for more information concerning fonts.

FONTLIB(libname):

Specifies the font library to be used when formatting for page printers or PostScript devices. The font library contains character-set information and image patterns of these characters. The font library requested on the SCRIPT command must be available on the system when the formatted document is printed.

Default font libraries differ depending of the page printer you are using.

For the 4250 Printer, the default font libraries are:

System	Default Library
ATMS-III	FONTLIB(FONT4250)
CMS	FONTLIB(FONT4250)
MVS/DLF	FONTLIB(FONT4250)

TSO FONTLIB(SYS1.FONT4250)
VSE/DLF FONTLIB(FONT4250)

For the 3800 Printing Subsystem Model 3 and Model 6, the default font libraries are:

System	Default Library
ATMS-III	FONTLIB(FONT38PP)
CMS	FONTLIB(FONT38PP)
MVS/DLF	FONTLIB(FONT38PP)
TSO	FONTLIB(SYS1.FONT38PP)
VSE/DLF	FONTLIB(FONT38PP)

For the IBM 3820 Printer, and the 4224-2xx Printer the default font libraries are:

System	Default Library
ATMS-III	FONTLIB(FONT3820)
CMS	FONTLIB(FONT3820)
MVS/DLF	FONTLIB(FONT3820)
TSO	FONTLIB(SYS1.FONT3820)
VSE/DLF	FONTLIB(FONT3820)

For PostScript devices, the default font libraries are:

System	Default Library
ATMS-III	Not supported by ATMS-III
CMS	FONTLIB(FONTTPS)
MVS/DLF	FONTLIB(FONTTPS)
TSO	FONTLIB(SCRIPT.R40.FONTTPS)
VSE/DLF	FONTLIB(FONTTPS)

For the LaserPrinter 4028, the default font libraries are:

System	Default Library
ATMS-III	FONTLIB(FONT300)
CMS	FONTLIB(FONT300)
MVS/DLF	FONTLIB(FONT300)
TSO	FONTLIB(SYS1.FONT300)
VSE/DLF	FONTLIB(FONT300)

For more information concerning fonts, see "Font Requirements for Page Printers" on page 126 and "Overriding the Default Font Using the CHARS Option" on page 127.

Using the SCRPTFNT DDNAME in TSO: An alternative method for specifying font libraries in TSO is to pre-allocate the desired libraries with a ddname of SCRPTFNT. When you have a ddname of SCRPTFNT allocated and you don't specify the FONTLIB command option, SCRIPT/VS searches all of the font libraries concatenated in SCRPTFNT. The libraries are searched in the order specified when you allocated SCRPTFNT.

When you have a ddname of SCRPTFNT allocated, the default font library is not added to the concatenation sequence. If you want the default font library to be searched, you must include it in the allocation of SCRPTFNT.

If you don't specify the FONTLIB command option and a ddname of SCRPTFNT is not allocated, the default font library is used.

If you specify the FONTLIB command option and a ddname of SCRPTFNT, SCRIPT/VS uses the FONTLIB option and ignores the SCRPTFNT ddname.

If you use a concatenated font library, the first DCFINDEX found in the concatenation must accurately represent all of the font members in the concatenation. For more information on creating a DCFINDEX with concatenated font libraries, refer to the *Document Composition Facility: SCRIPT/VS Text Programmer's Guide*.

FPASSES(n):

Allows you to specify more than two formatting passes on documents. When two formatting passes are inadequate to ensure that all page number references are resolved correctly, the FPASSES(n) option can be used to specify more passes. The IBM-supplied maximum number of formatting passes is 4 unless changed at your installation.

The FPASSES(n) option causes SCRIPT/VS to process your input document a specified number of times, and produces output only on the last pass. This option significantly increases computer processing time for the document.

Note: FPASSES and TWOPASS are mutually exclusive options. See the TWOPASS option.

INDEX:

Causes SCRIPT/VS to create an index from index entries specified in the text of the document. The :INDEX tag must be included in your source document at the point where you want the index placed—usually in the back matter. If the :INDEX tag is not a part of your source document, an index is not created.

LIB(name):

Specifies the names of the libraries containing SCRIPT/VS macros.

For the GML starter set, the APFs reside in macro libraries named DSMGML4 MACLIB (CMS environment) and SCRIPT.R40.MACLIB (TSO environment). In the ATMS-III environment, the APFs reside in SYSOP (system operator) permanent storage. These are the default library names; you need enter the LIB option only if you want a library other than, or in addition to, the default library.

NOSEGLIB:

Specifies that no library search is to be made for page segments. Use NOSEGLIB when you have specified page segments within your document with the .SI [Segment Include] control word, but those segments are not yet

within a segment library when you issue the SCRIPT command. This will prevent an error message from being issued. See "SEGLIB" option.

NOSORCDD:

This option overrides the SORCDD option and **explicitly** specifies that the primary input file is a data set name.

OPTIONS(name):

Specifies the name of a file containing more SCRIPT/VS command options. The options that are entered in the OPTIONS file are processed as if they had been entered with the SCRIPT command as separately listed options. Therefore, if you have a file (CMS) or a document (CICS) that contains the following lines

```
PROFILE(DSMPROF4)
BIND(6)
INDEX
SPELLCHK
```

in the CMS environment, you could enter the following

```
script mydoc (options(document))
```

instead of

```
script mydoc (profile(dsmprof4)
bind(6) index spellchk
```

Either of the above two SCRIPT commands would achieve the same results.

In the CICS environment, you could enter the following

```
script * (options('optdoc'))
```

instead of

```
script * (profile(dsmprof4)
bind(6) index spellchk
```

Either of the above two SCRIPT commands would achieve the same results.

Note: The OPTIONS option is available *only* in the CMS and CICS environments. The file type in CMS must be *OPTIONS*.

PAGE:

Allows you to print only the specific pages of formatted output that you want. The PAGE option has several formats, and any number of page range specifications can be included. If you use the prompting mode, that *replaces* the remainder of the suboptions. Valid forms of the page range specifications are:

```
PAGE [(PROMPT)]
```

```
PAGE [([FROM] frompage [TO] topage)]
```

```
PAGE [([FROM] frompage FOR n)]
```

```
PAGE [([FROM] page ONLY)]
```

If no suboption is given with the PAGE option, PAGE (PROMPT) is assumed (except under CICS, where it is ignored.)

PROFILE(name):

Specifies the name of a file or data set you want used as the document profile. The profiles available for use depend on the type of document you are processing.

The default profile provided with SCRIPT/VS for use with the GML starter set is:

CMS DSMPROF4 SCRIPT

TSO 'userid.DSMPROF4.TEXT'

ATMS-III DSMPROF4

Using the SCRPTPRO DDNAME in TSO: An alternative method for specifying the profile that you want SCRIPT/VS to use for processing in TSO is to pre-allocate the desired profile with a ddname of SCRPTPRO.

If you don't specify the PROFILE command option and SCRIPT/VS finds a ddname of SCRPTPRO allocated, SCRIPT/VS uses this ddname to locate the associated profile.

If the PROFILE command option is not specified and a ddname of SCRPTPRO is not allocated, SCRIPT/VS searches the default profile.

If you specify the PROFILE command option and a ddname of SCRPTPRO, SCRIPT/VS uses the PROFILE option and ignores the SCRPTPRO ddname.

Note: IEC141I 013-18 ABEND will occur if the member name specified on the DSNNAME parameter cannot be found.

PSOUT:

Specifies whether SCRIPT/VS generates PostScript output in ASCII or EBCDIC when you specify a PostScript device with the DEVICE option. If you specify a PostScript device and do *not* specify PSOUT, the default is to produce an ASCII file.

If you specify PSOUT(ASCII), SCRIPT/VS produces an ASCII file. You can then download the file to an IBM PC and print the file on a locally attached PostScript device, such as the IBM 4219 Personal Page Printer.

If you specify PSOUT(EBCDIC), SCRIPT/VS produces an EBCDIC file. You can look at the EBCDIC file (for diagnostic purposes) on the host system, but you cannot print the file.

Note: The PSOUT option is valid *only* for PostScript devices. It is ignored for non-PostScript devices.

SEGLIB:

Specifies the name of the segment library you want searched when formatting on page devices. SCRIPT/VS then searches the segment library for any page segments you have included in your document with the .SI [Segment Include] control word.

SCRIPT/VS automatically searches the default library for requested segments, but you must specify the SEGLIB option if the segment you request resides in a library you created. In general, SCRIPT/VS searches only one segment library, either the default library or the one you created, but not both. In TSO, you can pre-allocate more than one library with a ddname of SCRPTSEG to concatenate segment libraries.

Default segment libraries differ among the various page printers.

For the 4250 Printer, the default segment libraries are:

System	Default Library
ATMS-III	NOSEGLIB
CMS	SEGLIB(PSEG4250)
MVS/DLF	SEGLIB(PSEG4250)
TSO	SEGLIB(SYS1.PSEG4250)
VSE/DLF	(⁸)

For the 3800 Printing Subsystem Model 3 and Model 6, the default segment libraries are:

System	Default Library
ATMS-III	NOSEGLIB
CMS	SEGLIB(PSEG38PP)
MVS/DLF	SEGLIB(PSEG38PP)
TSO	SEGLIB(SYS1.PSEG38PP)
VSE/DLF	(⁸)

For the IBM 3820 Printer, 4224-2xx Printer, and LaserPrinter 4028, the default segment libraries are:

System	Default Library
ATMS-III	NOSEGLIB
CMS	SEGLIB(PSEG3820)
MVS/DLF	SEGLIB(PSEG3820)
TSO	SEGLIB(SYS1.PSEG3820)
VSE/DLF	(⁸)

Using the SCRPTSEG DDNAME in TSO: An alternative method for specifying segment libraries in TSO is to pre-allocate the desired libraries with a ddname of SCRPTSEG. When you have a ddname of SCRPTSEG allocated and you don't specify the SEGLIB command option, SCRIPT/VS searches all of the segment libraries concatenated in SCRPTSEG. The libraries are searched in the order specified when you allocated SCRPTSEG.

When you have a ddname of SCRPTSEG allocated, the default segment library is not added to the concatenation sequence. If you want the default segment library to be searched, you must include it in the allocation of SCRPTSEG.

If you don't specify the SEGLIB command option and a ddname of SCRPTSEG is not allocated, the default segment library is used.

⁸ For more information about the .SI [Segment Include] control word in the VSE environment, refer to the *Document Composition Facility: SCRIPT/VS Language Reference*.

If you specify the SEGLIB command option and a ddname of SCRPTSEG, SCRIPT/VS uses the SEGLIB option and ignores the SCRPTSEG ddname.

SORCDD:

An alternative method for specifying the primary input file in TSO is to use a pre-allocated ddname. To do this, pre-allocate the ddname and use it instead of the *file-id* on the SCRIPT command, followed by the command option SORCDD. The SORCDD command option notifies SCRIPT to process the primary input file as a ddname.

You can pre-allocate a ddname and then use the SORCDD command option as shown in the following example:

- Allocate the primary input data set with the TSO ALLOCATE command:

```
ALLOC DD(ABCD) DSN('userid.DATA.TEXT(FILEIN)') SHR REUSE
```

- To use the ddname ABCD as the primary input file, you need to use the command option SORCDD:

```
SCRIPT ABCD NOPROF CONT TWO SORCDD
```

Note: IEC141I 013-18 ABEND will occur if the member name specified on the DSNNAME parameter cannot be found.

SPELLCHK:

Causes SCRIPT/VS to perform spelling verification. Words that are not included in the SPELLCHK spelling dictionary are listed with the error messages.

SYSVAR(x value):

Specifies system variables that can be used to control the processing of a document. There can be one or more pairs of variable codes (x) and corresponding values.

The GML starter set uses several system variables to provide control over document processing. See "SYSVAR Option" on page 133.

TWOPASS:

Causes SCRIPT/VS to process your input document twice, producing output only on the second pass.

This is necessary to resolve forward references (as when an automatically generated table of contents or list of illustrations is in the front of a book).

Note: TWOPASS and FPASSES are mutually exclusive options. See the FPASSES(n) option.

Logical Device and Output Destination

DEVICE(type):

Specifies the *logical* device and underlying physical device for which formatting is to be performed. In this context, a logical device means a combination of physical device type, page size, and number of lines per vertical inch. The logical device types supported in the starter set are shown in Table 7 on page 128, Table 9 on page 129, and Table 8 on page 129. If you do not

specify a device, the default is TERM in the CMS and TSO environments, and 1403W6 in the batch environment.

FILE(name):

Causes output to be formatted for the specified logical device but stored in the named file or data set instead of being displayed or printed.

The FILE option is specified as:

FILE [(file-id)]

In CMS, the file-id is of the form:

filename [filetype [filemode]]

and the default file name is \$filenam, where filenam is the first seven characters of the input file name, preceded by a dollar sign (\$). The default file type is SCRIPT.

For page devices, the default file name is the eight characters of the input file name. The default file type is:

- **LIST4250** for the 4250 Printer.
- **LIST38PP** for the 3800 Printing Subsystem Model 3 or Model 6.
- **LIST3820** for the IBM 3820 Page Printer and for printers compatible at the data stream level.
- **LISTPS** for PostScript devices.
- **LIST4028** for the LaserPrinter 4028

The default file mode is A1 for all devices.

In TSO, the file-id is a fully or partially qualified data set name.

Using the SCRPTFIL DDNAME in TSO: An alternative method for specifying the direct-access file for the formatted output in TSO is to pre-allocate the desired direct-access file with a ddname of SCRPTFIL.

If you don't specify the FILE command option and SCRIPT/VS finds a ddname of SCRPTFIL allocated, SCRIPT/VS uses this ddname as the direct-access file for the formatted output.

If the FILE command option is not specified and a ddname of SCRPTFIL is not allocated, the existing DCF naming conventions apply. For more details on TSO data set naming conventions, refer to the *Document Composition Facility: SCRIPT/VS Language Reference*.

If you specify the FILE command option and a ddname of SCRPTFIL, SCRIPT/VS uses the FILE option and ignores the SCRPTFIL ddname.

Note: The FILE option is not available in the CICS environment except when formatting for page devices.

PRINT:

Causes output to be formatted for the specified logical device and printed on the system printer (in CMS and TSO); Files sent to AFP page printers will go directly to SPOOL saving processing time and DASD. If no logical device is specified, DEVICE(1403W6) is assumed.

Note: This option is ignored for 4250 printers and PostScript devices.

TERM:

Causes output to be formatted for the specified logical device but displayed at a terminal, instead of on the physical device. If no logical device is specified, DEVICE(TERM) is assumed.

Note: TERM is not available in the batch environment and is forced in the CICS environment.

Error Handling

CONTINUE:

Normally, SCRIPT/VS stops processing when it finds an error. This option directs it to continue processing unless the error is severe.

MESSAGE(DELAY):

This option, with the DELAY parameter specified, causes error messages to be printed at the end of the output document instead of being displayed at a terminal during processing. (In the batch processing environment, messages are always delayed.)

Font Requirements for Page Printers

SCRIPT/VS requires special fonts to produce output for page printers. The fonts for these devices must be installed and made available in a font library, and must be known to DCF by being included in a DCFINDEX. (Refer to the appendix "The DCF Font Library Index Program" in *Document Composition Facility: SCRIPT/VS Text Programmer's Guide* for more information.)

You can modify DCF or use the CHARS option of the SCRIPT command to select a different default typeface family. See "Overriding the Default Font Using the CHARS Option" on page 127 for more information.

The defaults chosen by SCRIPT/VS require that the following font licensed programs be installed:

- For the IBM 4250 Printer:
 - 5771-AAR Monotype Times New Roman
 - 5771-AAW Typewriter and Pi Specials.
- For the IBM 4224 Printer, SCRIPT/VS requires the Courier and Essay typeface families. These fonts are included with PSF Version 1 and 2.
- For the IBM LaserPrinter 4028, SCRIPT/VS requires the Courier and Times Roman typeface family font metrics. These fonts are available in PSF Version 2 (MVS and VM).
- For other IBM AFP page printers (such as the 3820):
 - 5771-ABA Sonoran Serif
 - 5771-ABC PI and Specials; and the Prestige Typeface family included with the Print Services Facility and Print Management Facility licensed programs.

Note: The OCRB font is recommended for printing human-readable information with bar codes, but is not required.

Font Requirements for PostScript Devices

SCRIPT/VS requires that the following typeface families be available to PostScript devices in order to format GML starter set documents:

- Times Roman
- Courier.

Each font that is used in the document must be available to the PostScript device and must have a corresponding Adobe Font Metrics (AFM) file in the PostScript font library. For example, to format a GML starter set document with Times Roman and Courier fonts, the Times Roman and Courier AFM files must be available in the PostScript font library.

For more information about PostScript fonts, see the *Document Composition Facility: SCRIPT/VS User's Guide*.

Overriding the Default Font Using the CHARS Option

If you want to override the default typeface when formatting for a page printer or a PostScript device, specify a different font in the CHARS option of the SCRIPT command. Although you can specify up to 31 fonts on CHARS, the starter set uses only the first one specified.

The initial font, specified either by the default font or in the CHARS option, must be a typographic font available in a variety of sizes and weights. See your document administrator to learn which fonts are available at your location and how to access them. Also, refer to the *Document Composition Facility: SCRIPT/VS Text Programmer's Guide* for more information on using the CHARS option.

Issuing the *SCRIPT* Command

The *SCRIPT* command must be issued differently, depending on the environment you use, so each environment is described separately. Consult your document administrator for the actual operating procedures for your installation. These could include combinations of system procedures (that is, EXECs, CLISTs, PROCLIBs, or JCL) and manual operating procedures.

When issuing the *SCRIPT* command in the CMS environment, you must separate the options from the file identifier by a left parenthesis.

SCRIPT/VS Logical Devices

These tables list the logical devices that can be specified with the *DEVICE* option of the *SCRIPT* command and the default page dimensions for each. These are the defaults supplied by IBM. You can change the defaults at installation time. The page size can be changed with the .PW [Page Width] and .PL [Page Length] control words. The page margins can be changed with the .PM [Page Margins], .AM [Adjust Margin], .TM [Top Margin], and .BM [Bottom Margin] control words.

SCRIPT/VS Logical Line Devices

Logical Device Type	Intended Output Device	Lines per Inch	Page Size		Margins			Line Length
			Width	Length	Bind	Top	Bottom	
TERM 2741 3270	(1) 2741 3270	6	132 204	11i 11i	2	.5i	.5i	6i
1403N6 1403N8 1403W6 1403W8 1403W6S 1403W8S 1403SW ² STAIRS	1403	6 8 6 8 6 8 6 6	8.5i 8.5i 13.5i 13.5i 13.5i 13.5i 8.5i 13.5i	11i 11i 11i 11i 8.5i 8.5i 11i 11i	1i	.5i	.5i	6i
3800N6 3800N8 3800N12 3800W6 3800W8 3800W12 3800N6S 3800N8S 3800N12S 3800W6S 3800W8S 3800W12S	3800	6 8 12 6 8 12 6 8 12 6 8 12	8.5i 8.5i 8.5i 13.5i 13.5i 13.5i 11i 11i 11i 13.5i 13.5i 13.5i	10i 10i 10i 10i 10i 10i 7.5i 7.5i 7.5i 7.5i 7.5i 7.5i	1i	0	0	6i

1

The physical device type corresponding to the TERM logical device can be either 2741 or 3270, depending upon the actual terminal type. Under ATMS, the physical device type is always 2741.

2

This is a 12-pitch device; all other 1403 devices are 10-pitch.

Table 7. *SCRIPT/VS* Logical Line Devices. This table lists the logical line devices that can be specified with the *DEVICE* option of the *SCRIPT* command and the default page dimensions for each device.

SCRIPT/VS Logical PostScript Devices

Logical Device Type	Intended Output Device	Page Size		Margins			Line Length
		Width	Length	Bind	Top	Bottom	
PSA	PostScript	8.5i	11i	1i	.5i	.5i	6i
PSA90		11i	8.5i				
PSA180		8.5i	11i				
PSA270		11i	8.5i				
PSB		11i	17i				
PSL		8.5i	14i				
PSA3		297mm	420mm				
PSA4		210mm	297mm				
PSB5		182mm	257mm				

Table 8. SCRIPT/VS Logical PostScript Devices. This table lists the logical PostScript devices that can be specified with the DEVICE option of the SCRIPT command and the default page dimensions for each device. The line spacing for PostScript logical devices is determined by the .LS [Line Spacing] control word and the fonts used in the document.

SCRIPT/VS Logical Page Devices

Logical Device Type	Intended Output Device	Page Size		Margins			Line Length
		Width	Length	Bind	Top	Bottom	
38PPN	3800-3	8.5i	10i	1i	0	.125i	6i
38PPW		13.5i	10i				
38PPNS		11i	7.5i				
38PPWS		13.5i	7.5i	.5i	.5i	.5i	6i
38PPW90		10i	13.5i				
38PPNS90		7.5i	11i				
38PPW270		10i	13.5i				
3820A	3820 ¹	8.5i	11i	1i	.5i	.5i	6i
3820A90		11i	8.5i				
3820A180		8.5i	11i				
3820A270		11i	8.5i				
3820L		8.5i	14i				
3820A4		210mm	297mm				
3820B4		257mm	364mm				
3820B5		182mm	257mm				
4250A	4250	8.5i	11i	1i	.5i	.5i	6i
4250B		11i	17i				
4250L		8.5i	14i				
4250A3		297mm	420mm				
4250A4		210mm	297mm				

¹ The DCF physical device type specified is 3820. The actual device can be any AFP printer compatible with the 3820 at the data stream level.

Table 9. SCRIPT/VS Logical Page Devices. This table lists the logical page devices that can be specified with the DEVICE option of the SCRIPT command and the default page dimensions for each device. The line spacing for these is determined by the .LS [Line Spacing] control word and the fonts used in the document.

SCRIPT/VS Logical Page Devices (cont.)

Note: Mixing different fonts in a document may cause problems.

Logical Device Type	Intended Output Device	Page Size		Margins			Line Length
		Width	Length	Bind	Top	Bottom	
PG1A PG1A90 PG1A180 PG1A270 PG1L PG1A4 PG1B4 PG1B5	3820 ¹	8.5i 11i 8.5i 11i 8.5i 210mm 257mm 182mm	11i 8.5i 11i 8.5i 14i 297mm 364mm 257mm	li	.5i	.5i	6i
PG2A PG2A90 PG2A180 PG2A270 PG2L PG2A4 PG2B4 PG2B5	3812 3816	8.5i 11i 8.5i 11i 8.5i 210mm 257mm 182mm	11i 8.5i 11i 8.5i 14i 297mm 364mm 257mm	li	.5i	.5i	6i
PG3A PG3W PG3NS PG3L PG3A4 PG3B4 PG3B5	4224 ²	8.5i 13.5i 11i 8.5i 210mm 257mm 182mm	11i 11i 8.5i 14i 297mm 364mm 257mm	li	.5i	.5i	6i
PG4A PG4A90 PG4A180 PG4A270 PG4L PG4A4 PG4B5 PG4E	4028	8.5i 11i 8.5i 11i 8.5i 210mm 182mm 184.2mm	11i 8.5i 11i 8.5i 14i 297mm 257mm 266.7mm	li	.5i	.5i	6i
¹ The DCF physical device type specified is 3820. The actual device can be any AFP printer compatible with the 3820 at the data stream level. ² The DCF physical device type specified is 4224. The actual device can be any printer compatible with the 4224 at the data stream level.							

Table 10. SCRIPT/VS Logical Page Devices (cont.). This table lists the logical page devices that can be specified with the DEVICE option of the SCRIPT command and the default page dimensions for each device. The line spacing for these is determined by the .LS [Line Spacing] control word and the fonts used in the document.

CMS Environment

The Conversational Monitor System (CMS) component of VM/370 is a single-user operating system that provides interactive file creation and manipulation facilities that can be useful for document development. SCRIPT/VS uses the input/output facilities of CMS to read source documents and to write the formatted output. Processing output can be directed to your CMS terminal, a VM/370 printer, or a CMS file for later use.

The APFs and a profile must be available to CMS before you invoke SCRIPT/VS to process your document. A default file type of SCRIPT is assumed for all files if a file type is not specified.

In the following example:

```
script techrprt ( prof(dsmprof4) two bind(1i)
```

- techrprt is the file name of the document to be processed. The default file type of SCRIPT is assumed.
- The PROFILE option causes SCRIPT/VS to use the file DSMPROF4 SCRIPT as the profile.
- The TWOPASS command option causes SCRIPT/VS to make two formatting passes through this file before formatting it for output.

Note: You can also use the FPASSES option of the SCRIPT command to specify two or more formatting passes.

- The BIND option causes SCRIPT/VS to shift the printed output one inch to the right on all pages.
- The logical device for which the output is formatted is TERM because the device option was not specified. The destination is the same as the logical device because no destination was specified. The results are therefore displayed at your terminal.

In the next example:

```
script mydoc ( mess(delay) prof(dsmprof4) print spell
```

- mydoc is the file name of the document to be processed. The default file type of SCRIPT is assumed.
- The MESSAGE option causes error messages to be appended to the output instead of being displayed at the terminal during processing.
- The PROFILE option causes SCRIPT/VS to use the file DSMPROF4 SCRIPT as the profile.
- The PRINT option tells SCRIPT/VS to direct output to the system printer. Because no logical device was specified, DEVICE(1403W6) is assumed.
- The SPELLCHK option causes SCRIPT/VS to verify the spelling of the words in the document. Words not found in the spelling dictionary are included with the messages.

TSO Environment

The Time Sharing Option (TSO) is a processor available to OS/VS2-MVS users. SCRIPT/VS uses the input/output facilities of TSO to read source documents and to write the formatted output. Processing output can be directed to a TSO terminal, an MVS printer, or a TSO data set for later use.

APFs and a profile data set must be available to TSO before you invoke SCRIPT/VS to process your document.

All data set names that are not fully qualified assume "userid." and ".TEXT" as default qualifiers.

In the following example:

```
script techrprt(rpt1) prof(dsmprof4) two bind(10cm) page(5 for 4)
```

- techrprt is the name of a partitioned data set whose member rpt1 is the document to be processed. The default qualifiers of "userid." and ".TEXT" are assumed, producing the following fully qualified name:

userid.TECHRPT.TEXT(RPT1)

- The PROFILE option causes SCRIPT/VS to use the file userid.DSMPROF4.TEXT as the profile.
- The TWOPASS command option causes SCRIPT/VS to make two formatting passes through this file before formatting it for output.
Note: You can also use the FPASSES option of the SCRIPT command to specify two or more formatting passes.
- The BIND option causes SCRIPT/VS to shift the printed output 10 centimeters to the right on all pages.
- The PAGE option causes SCRIPT/VS to print only pages 5, 6, 7, and 8 of the formatted document.
- The logical device for which the output is formatted is TERM because the device option was not specified. The destination is the same as the logical device because no destination was specified. The results are therefore displayed at your terminal.

In the next example:

script 'userid.mydoc' cont prof(dsmprof4) print spell

- userid.mydoc is the fully qualified data set name of the document to be processed.
- The CONTINUE option causes SCRIPT/VS to continue processing even after an error is found, unless the error is a severe one.
- The PROFILE option causes SCRIPT/VS to use the file userid.DSMPROF4.TEXT as the profile.
- The PRINT option tells SCRIPT/VS to direct output to the system printer. Because no logical device was specified, DEVICE(1403W6) is assumed.
- The SPELLCHK option causes SCRIPT/VS to verify the spelling of the words in your document.

ATMS-III System

The Advanced Text Management System-III (ATMS-III) is a program product that runs with CICS/VS to OS/VS2 MVS and VSE users. SCRIPT/VS uses the input/output facilities of ATMS-III to read source documents and to write the formatted output. Processing output can be directed to any ATMS-III output device or held for review at a terminal.

APFs and a profile document must be available to ATMS-III before you invoke SCRIPT/VS to process your document.

The source document must be in working storage when SCRIPT/VS is invoked to process it.

Commands issued in the ATMS-III environment must be entered in lower-case letters.

In the following example:

```
script * prof(dsmprof4) cont index
```

- * indicates that the document to be processed resides in the operator's working storage.
- The PROFILE option causes SCRIPT/VS to use the document DSMPROF4 as the profile.
- The CONTINUE option causes SCRIPT/VS to continue processing even after an error has been found unless the error is severe.
- The INDEX option causes SCRIPT/VS to create a subject index from index entries specified in the document.
- The logical device for which the output will be formatted is TERM. After SCRIPT/VS processing has been completed, the formatted output can be reviewed at a terminal.

Batch Processing

You can run SCRIPT/VS in a batch environment (MVS or VSE) by either using the Document Library Facility (DLF) program product or by running it as a batch TSO job. Processing output can be directed to a system printer or to a document data set.

Information about DLF can be found in the *Document Library Facility Guide*. Consult your document administrator for information about the batch processing procedures in use in your installation.

SYSVAR Option

The DSMPROF4 profile is provided for use with the GML starter set for formatting general documents. It produces formatted output on terminals, line printers, and page printers.

You can use the SYSVAR option of the SCRIPT command, described above, to control several aspects of document formatting. The SYSVAR option is entered like any other option, with its values enclosed in parentheses.

The recognized SYSVAR variables are:

- D Duplex:** Indicates whether major sections, such as the front matter, body, appendixes, and sections started with level-0 and level-1 headings (:H0 and :H1) should begin on odd-numbered pages with the text of the level-0 and level-1 headings right justified for one- and two-column style. Recognized values are:

YES Format for duplexing.

NO Do not format for duplexing.

The default is no; new sections can begin on odd- or even-numbered pages.

- H Head Numbering:** Indicates whether level one through four headings (:H1 through :H4) are to be automatically numbered in the body of the document. Recognized values are:

YES Number level-one through level-four headings.

NO Do not number headings.

initial-value The initial setting of the heading counters. By default, headings are numbered from 1.0.

The default is no; headings are not numbered.

P Process: Lets you specify the process being run. The names of the physical output device and the logical device are always used as active processes for :PSC elements. See ":PSC—Process-Specific Control" on page 85 for details.

R Read: Reads a file created earlier with SYSVAR W; for example

`sysvar (r myids)`

The named file holds the collected IDs from the document processed when SYSVAR W was specified with the file name MYIDS. Reading that same file (using SYSVAR R) resolves forward references to the IDs within the document, so that page references resolve correctly.

SYSVAR R (for READ) and W (for WRITE) can both be specified using the same name. If, for example, you entered:

`sysvar (r myids w myids)`

changes made to IDs, or IDs added to the document being processed, are included in a new file (specified with the SYSVAR W) while the earlier version (specified with the SYSVAR R) is being read.

Once the ID file has become stabilized and no new ID entries are being made, a document can be formatted with correctly resolved forward references with only one pass necessary when SYSVAR R is used. The use of FPASSES will achieve the same result. See the section on FPASSES(n) under "SCRIPT Command Options" on page 117.

Note: SYSVAR R is valid *only* in CMS and TSO.

S Style: Controls the page style of the body of the document. Recognized values are:

ONE Single-column layout

TWO Double-column layout

OFFSET Single-column layout with outdented headings

The default is single-column layout.

T Title Page Printing: Indicates whether a title page should be generated from the information collected by the title page group tags. Recognized values are:

YES Create a title page and align the text against the right margin.

RIGHT Create a title page and align the text against the right margin.

CENTER Create a title page and center the text.

LEFT Create a title page and align the text against the left margin.

NO Do not create a title page.

The default is right aligned, and a title page will be created if the title page tags are specified in the document. Figure 10 on page 113 illustrates the title page for this document.

W Write: Creates a file at the end of the last pass, using the name specified. For example,

```
sysvar (w myids)
```

creates a file, MYIDS, that contains all the IDs (except the indexing IDs) included in the document being processed. When this same file is read (using the SYSVAR R option), the collected IDs are used to resolve forward references in a document so that page references resolve correctly.

SYSVAR R (for READ) and W (for WRITE) can both be specified using the same name. If, for example, you entered:

```
sysvar (r myids w myids)
```

changes made to IDs, or IDs added to the document being processed, are included in a new file (specified with the SYSVAR W) while the earlier version (specified with the SYSVAR R) is being read.

Note: SYSVAR W is valid *only* in CMS and TSO. For information about specifying multiple formatting passes with the TWOPASS or FPASSES option of the SCRIPT command, see the sections on both the FPASSES(n) and TWOPASS options of the SCRIPT command under "SCRIPT Command Options" on page 117.

X Cross-Referencing: Indicates whether a cross-reference listing of figure, table, footnote, heading, list item, and index IDs, and imbedded files should be appended to the formatted output. Recognized values are:

YES Create ID and imbed file cross-references.

NO Do not create cross-references.

The default is yes; cross-reference listings are appended to the formatted output.

For example, the following command causes output to be printed in two columns with headings numbered from 4.0, all control words entered between :PSC PROC=TUNE and :EPSC in the source document are processed, the output is not duplexed, no index will be created, and the text of the title page is right-aligned. The CMS syntax is shown. For TSO and CICS, the initial left parenthesis is omitted.

```
script mydoc ( profile(dsmprof4) sysvar(s two h 4.0 p tune)
```

Variations other than the heading numbering, number of columns, and processing the control words within the :PSC tag were defaults and were not specified in the command.

Note: Consult your document administrator for other profiles and variations that may have been established; such as for double-spaced drafts or for pre-processing documents for use with other text processors. Your document administrator may also have prepared EXECs, CLISTs, procedures, or

OPTIONS files for specific tasks to make it unnecessary to specify the individual profile and variations.

Interim Processing

This manual explains how to mark up a general document in its *normal* form. This is the form in which it is processed for final printing and in which it remains in your computer files for any processing that might be required in the future. Profiles and APFs are primarily designed to work correctly when a document is in the normal form.

During the course of production, though, your document might not always be in the normal form. At times it will be incomplete or incorrectly marked up. Or, you might wish to process only a part of the document—perhaps a heading segment that was just revised. The starter set profile and APFs allow you considerable flexibility during the production cycle:

- You can temporarily enter the :TOC, :FIGLIST, and :TLIST tags as the last tags in the back matter, thereby obtaining a table of contents, a list of illustrations, and a list of tables without the expense of multiple formatting passes. Entering these items in the back matter will, however, cause them to be in two column format. (Note that you may want the second pass anyway, if you need to check the resolution of cross references.)
- Parts of the document can be processed independently, although results might not be identical to those you will obtain when the entire document is processed in the normal form.

You can improve these interim results by including some of the higher-level tags in the processing run. For example, if you want to process only a second-level heading segment, you can obtain better output if you insert a :GDOC, :BODY, or :H1 tag ahead of it, because these APFs will establish the correct page layout, running headings, heading numbering (if requested), and so on.

- In CMS and TSO, when you process parts of a document, you can have correctly resolved cross-references to the entire document by using the SYSVAR R option on the SCRIPT command. The file previously created with the SYSVAR W option when the entire document was processed contains all the IDs for that document. When you specify that file with the SYSVAR R option, all the IDs (except indexing IDs) are available for cross-referencing.

Note: SYSVARs R and W are valid *only* in CMS and TSO.

- Incomplete or inaccurate markup can cause errors that would normally persist for the remainder of the document. For example, failure to terminate a list might cause succeeding text to be indented. To minimize the impact of such errors, some APFs perform *housekeeping* functions in addition to their normal processing. For example, the heading APFs terminate any current lists. These safeguards make it more likely that your draft copies will appear to be correct; however, you should correct the markup to eliminate the error conditions.

Markup Errors

There are two kinds of markup error:

- Entering markup you think is right, but isn't
- Making a typographical error that causes SCRIPT/VS to misinterpret markup (or to treat as markup something that is not).

You can prevent the first type of mistake by understanding this manual and by taking care in identifying the elements of your document. The second type of mistake obviously can be avoided by careful typing.

Of course, no one is perfect, and errors will occur. The next best thing to avoiding errors is being able to recognize where they are and to correct them without difficulty. In computer text processing, finding and correcting markup errors is called *debugging* your markup.

Some common markup *bugs* are as follows:

- Omitting the period at the end of markup (when required)
- Omitting the period at the end of a symbol
- Omitting the ending tag of a list, figure, example, table, or other element that requires explicit termination
- Omitting the :P tag before the first paragraph after a level-five or level-six heading
- Using tags where they are not permitted, such as using a highlighted phrase tag within the text of a heading tag
- Using control words that conflict with the control words in the tag APFs.

For example, you might find that entire pages of your output are underscored. This could be caused by omitting a highlighted phrase end-tag. You might have actually forgotten to include the tag, or you might have misspelled it. In the latter case, SCRIPT/VS would not know it was a tag and would treat it as text. This bug is easy to find: just go to the point where the underscoring began and start looking for the place where it should have ended.

It is important to realize that even though dozens of words (or even dozens of pages) may look wrong in the output, it does not necessarily mean you must make dozens of corrections in the source. In fact, you will often find that the worse your output looks, the easier it will be to locate the bugs and correct them.

Sometimes an APF can determine that a markup error was made. In such cases, it will take some appropriate processing action and issue a message to help you correct the error. Refer to the *Document Composition Facility Messages* for information on the text of messages issued by the starter set APFs, together with explanations.

You can also receive error messages from SCRIPT/VS itself. Consult your document administrator about these.

Appendix A. The DSMPROF4 Profile

The DSMPROF4 profile is provided with SCRIPT/VS for use with the GML starter set. This profile ensures that the correct release of SCRIPT/VS is being used and that the DSMGML4 macro library is available. Then it does the following:

- Establishes the initial tag-to-APF mappings for the starter set tags
- Establishes formatting parameters
- Describes use of the epifile

The DSMGML4 macro library, described in Appendix B, "The DSMGML4 Macro Library" on page 145, together with the DSMPROF4 profile, constitute the GML starter set provided with SCRIPT/VS.

Modifications to the GML starter set, exactly as they are described in this chapter, are valid. Any problems arising from changes that are made as described here are eligible for service maintenance. Other modifications, of course, can not conflict with default or specified parameters because they may cause undesirable results. For example, you cannot set the indentation of list items so that it extends beyond column line length, nor can you set the indentation of examples so that example lines are truncated. For additional modification possibilities, refer to the *Document Composition Facility: Generalized Markup Language Starter Set Implementation Guide*.

GML Tag-to-APF Mapping

The DSMPROF4 profile maps the tags recognized by the starter set to APFs contained in the DSMGML4 macro library, using the SCRIPT/VS .AA [Associate APF] control word. The mapping established in the DSMPROF4 profile is often only an initial association; a number of tags change the tag-to-APF mappings of other tags. The .AA control word is also used to establish whether the tag will recognize attributes. Refer to the *Document Composition Facility: SCRIPT/VS Text Programmer's Guide* for details about .AA and other control words.

Formatting Parameters

The DSMPROF4 profile defines a variety of parameters that are used by the starter set in formatting document elements.

See Figure 11 on page 140. This figure shows what the settings are for indentation, skipped space, and highlighting levels for the document elements discussed in this section. The information in the figure is taken directly from DSMPROF4 and can be changed within the profile for permanent changes to the initial settings.

The following symbols define the amount of white space and indentation surrounding various kinds of text:

```
&@in@d &@sk@d - definition list terms
&@in@f &@sk@f - figures
&@in@o &@sk@o - ordered list items
&@in@p &@sk@p - paragraphs
&@in@q &@sk@q - long quotes
&@in@s &@sk@s - simple list items
&@in@u &@sk@u - unordered list items
&@in@x &@sk@x - examples
&@in@g &@sk@g - glossary definitions
&@in@z &@sk@z - list items
      &@sk@n - footnote
&@in@c &@ir@c - cell
&@in@t &@sk@t - table
```

The following symbols define various highlighting levels

```
&@hi@h - definition list headings
&@hi@d - definition list terms
&@hi@g - glossary list terms

.gs args 10      2      4      4      0      3      4      4      2      0
.gs vars @in@d @in@f @in@z @in@o @in@p @in@q @in@s @in@u @in@x @in@g

.gs args .75    1      .75    .75    .75    1      .75    .75    1      .75
.gs vars @sk@d @sk@f @sk@z @sk@o @sk@p @sk@q @sk@s @sk@u @sk@x @sk@g

.gs args .li    .li    .li    .li
.gs vars @sk@t @in@c @ir@c @in@t

.gs args .75    1      3      2      2
.gs vars @sk@ls @sk@n @hi@h @hi@d @hi@g
```

Figure 11. Symbols and Initial Formatting Parameters Set in DSMPROF4

The first line in the figure preceded by ".gs args" indicates the indentation for each of the variables just below it. The second line preceded by ".gs args" indicates the number of lines to be skipped before the variable just below it. The third line preceded by ".gs args" indicates the number of lines to be skipped before and after a table, and the left and right indentation values of text in a cells of a table. The fourth line preceded by ".gs args" indicates the number of lines to be skipped before and after lists and before a footnote. The next three numbers and variables are for highlighting levels. (.gs args and .gs vars are just quick ways to set many symbols in one place.)

The following paragraphs explain the variables depicted in the figure.

Examples: The symbols &@in@x and &@sk@x define the indentation of and white space preceding examples, respectively.

If you want to change the indentation of examples in just one document, you can set a symbol at the beginning of that document that looks like this:

```
.se @in@x = 4
```

All the examples in your document would then be indented 4 spaces instead of the 2 as set in DSMPROF4.

Similarly, if you want to change the number of lines skipped before an example from the setting in DSMPROF4 (which is 1, as shown in Figure 11 on page 140), you could set a symbol at the beginning of your document that looks like this

```
.se @sk@x = 2
```

and each example would be preceded by 2 blank lines rather than 1.

Figures: The symbols `&@in@f` and `&@sk@f` define the indentation of and white space preceding figures, respectively. The symbols `&@figplace` and `&@figwidth` give the default values of the PLACE and WIDTH attributes, respectively.

If you want to change the indentation of figures in just one document, you can set a symbol at the beginning of that document that looks like this:

```
.se @in@f = 3
```

All the figures in your document would then be indented 3 spaces instead of the 2 as set in DSMPROF4.

Similarly, if you want to change the number of lines skipped before a figure from the setting in DSMPROF4 (which is 1, as shown in Figure 11 on page 140), you could set a symbol at the beginning of your document that looks like this

```
.se @sk@f = 2
```

and each figure would be preceded by 2 blank lines rather than 1.

If you want to change the default placement of figures from its setting in DSMPROF4 of TOP, you could set the following symbol at the beginning of your document:

```
.se @figplace = inline
```

or

```
.se @figplace = bottom
```

If you want to change the default width of figures within your document from its setting in DSMPROF4 of PAGE, you could set the following symbol at the beginning of your document:

```
.se @figwidth = column
```

If you want to make a permanent change to DSMPROF4, you could simply change the ".gs args" setting in the profile that appears above `@figplace` and `@figwidth`.

Headings: The symbol &@bodyhead1 can be set to a string that will prefix each level-one heading in the body of the document. This can be done regardless of whether or not headings are numbered. (See the SYSVAR H option). You could, for example, set &@bodyhead1 to prefix the text of each level-one heading with the word "Chapter" by setting this symbol at the beginning of your document:

```
.se @bodyhead1 = Chapter
```

Each level-one heading in the **body** of the document would be preceded by the word "Chapter" and would be serially numbered beginning with 1.

Tables: The symbol &@in@t defines the left and right indentation for the table caption and description. The symbol &@sk@t defines white space preceding and following tables. The symbols &@in@c and &@ir@c define the left and right indentation in cells in a table, respectively.

If you want to change the left indentation of cells in a table in just one document, you can set a symbol at the beginning of that document that looks like this:

```
.se @in@c = 2
```

The cells in tables in your document will be indented on the left 2 spaces instead of the tenth of an inch as set in DSMPROF4.

You might want to decrease the left and right indentation values when your row is comprised of many cells. In this case, the tenth of an inch default indentation can be too large. Similarly, if you format for a wider-than-normal column line length, with only a few cells in the row, the tenth of an inch might not be enough indentation. You could set these symbols to be more than one tenth of an inch.

If you want to make the rules around the table a little thicker, redefine the named rule @t@thick:

```
.dr @t@thick weight .7mm
```

If you want the rules around the table to be the same thickness as the rules around the rows, change the value of the following symbols:

```
.se @thrules = boxrule  
.se @tvrules = boxrule
```

Hyphenation: Hyphenation is performed with the default spelling verification and hyphenation dictionary. Hyphenation is attempted only if a word contains at least five characters. There must be at least two characters left on a line before a word will be hyphenated, and at least three characters of the hyphenated word must remain to be carried to the next line. No more than two successive lines will be hyphenated.

Lists: The symbol &@hi@d defines the default highlight level of definition list terms. The symbol &@hi@g defines the default highlight level of glossary list terms. These values can be overridden by the TERMHI attribute of the :DL tag or the :GL tag.

The symbol &@hi@h defines the default highlight level of definition list headings. This value can be overridden by the HEADHI attribute of the :DL tag.

The symbols &@olistnest and &@ulistnest define the sequence of identifiers that are associated with each nested level of ordered and unordered list items, respectively. Each symbol can be set to a string of numbers indicating the order of the identifiers to be used for each level of nesting. Figure 12 shows the range and type of item identifier each number implies.

Unordered Lists:					
For Line Devices	For PostScript Devices	For PG2 Generic Devices	For PG3 Generic Devices	For PG4 Generic Devices	For PG1 & all other Page Devices
1: •	•	•	•	*	•
2: —	•	■	■	—	■
3: —	•	•	•	*	▲
4: ■	•	■	■	—	△
5: •	•	•	•	*	□
Ordered Lists:					
1:	1.	2.	3.	4.	5. ...
2:	a.	b.	c.	d.	e. ...
3:	1)	2)	3)	4)	5) ...
4:	a)	b)	c)	d)	e) ...
5:	i.	ii.	iii.	iv.	v. ...
6:	i)	ii)	iii)	iv)	v) ...
7:	I.	II.	III.	IV.	V. ...
8:	A.	B.	C.	D.	E. ...

Figure 12. Item Identifiers for Ordered and Unordered Lists. The identifier numbers that can be used in the symbols &@olistnest and &@ulistnest and the types of identifiers they produce, are shown.

If you want to change the order in which list item identifiers appear in your nested ordered lists, set the symbol like this:

```
.se @olistnest = 7814
```

If you look at the values set as shown in Figure 12, you will see that 7814 translates as follows:

```
7 I. II. III. IV. ...
```

```
8 A. B. C. D. ...
```

```
1 1. 2. 3. 4. ...
```

```
4 a) b) c) d) ...
```

The symbol as set above has, in fact, changed the list item prefixes to the prefixes used for an outline.

The following symbols define the amount of indentation applied to various types of lists:

```
&@in@d - definition lists
&@in@o - ordered lists
&@in@s - simple lists
&@in@u - unordered lists
&@in@g - glossary lists
```


The following symbols define the amount of vertical white space preceding list items of various types of lists:

&@sk@d - definition lists
&@sk@o - ordered lists
&@sk@s - simple lists
&@sk@u - unordered lists
&@sk@g - glossary lists

These values can be overridden by the COMPACT attribute of the list tags.

The @sk@ls symbol controls the amount of space at the beginning and end of the list.

Any of the above symbols can be changed in the same manner as previously explained in "Examples" and "Figures."

Long Quotations: The symbols &@in@q and &@sk@q define the indentation and white space, respectively, surrounding long quotations. Either of the above symbols can be changed in the same manner as explained in "Examples" and "Figures."

Paragraphs: The symbols &@in@p and &@sk@p define the indentation and white space, respectively, preceding paragraphs.

Either of the above symbols can be changed in the same manner as explained in "Examples" and "Figures."

Epifile

The DSMPROF4 profile calls a macro to do *epifile* processing (an epifile is the second portion of a profile and is processed after the main document has been processed). It appends to your document a cross-reference listing of all heading, figure, footnote, list item, and index-entry IDs used in the document, plus a list of all imbedded files. The cross-reference listing can be suppressed by the SYSVAR X option.

In CMS and TSO the epifile can also create a file of the IDs used in the document for figures, headings, list items, and footnotes. The file name must have been specified with the SYSVAR W option of the command. The file type will be DSMREFS in CMS. The third qualifier of the file in TSO will be DSMREFS. The file is defined internally using a DD name of DSMUTREF.

The cross-reference listing and ID file are generated by the EGDOC tag. If there is no EGDOC tag, they will be generated by the epifile processing, provided DSMPROF4 was the profile named on the PROFILE option. But if you imbed DSMPROF4 in your profile and don't include an EGDOC tag, you will not get a cross-reference listing. For more information about DSMPROF4 epifile processing, refer to the *Document Composition Facility: Generalized Markup Language Starter Set Implementation Guide*, (SH35-0050).

Appendix B. The DSMGML4 Macro Library

The DSMGML4 macro library contains the APFs and supporting macros that implement the GML starter set and bar code tag set. It is intended for use with SCRIPT/VS Release 4.0 for processing documents marked up with Release 3 and above.

The DSMPROF4 profile, described in Appendix A, "The DSMPROF4 Profile" on page 139, together with the DSMGML4 macro library, constitute the GML starter set provided with SCRIPT/VS. The profile and library are discussed in detail in the *Document Composition Facility: Generalized Markup Language Starter Set Implementation Guide*.

Naming Conventions

The members of the DSMGML4 macro library can be subdivided into three categories:

- Application processing functions (APFs), which provide processing for the GML tags that compose the starter set. All APFs begin with DSM. The initial tag-to-APF mapping is performed in the DSMPROF4 profile.
- Attribute APFs, which provide processing for attributes of GML tags and are invoked by tag APFs using the .GS EXATT control word. The names of these macros all begin with DSM@.
- Internal macros, invoked internally for various reasons. The names of these macros all begin with DSM#.

The GML starter set makes extensive use of the symbol-processing capabilities of SCRIPT/VS. Most symbols defined by the starter set begin with an "at" sign (@) to distinguish them from user-defined symbols. The DSM@MAC@ symbol contains the name of the macro library used by DSMPROF4 to verify that the correct macro library is being used.

Application Processing Functions

Major Document Parts

The tags that identify the major parts of the document are processed by the DSMFRONT, DSMBODY, DSMAPPD, and DSMBACKM macros. Each of these macros calls the DSM#RSET, DSM#DUPL, and DSM#STYL macros:

- The `DSM#RSET` macro checks for open quoted phrases, lists, examples, figures, and footnotes. If any of these elements is found to be open, a message is issued and the element is ended.
- The `DSM#DUPL` macro advances the formatted output to the next page, or, if duplexing has been requested with the `SYSVAR D` option, to the next odd-numbered page.
- The `DSM#STYL` macro establishes the page layout for the following pages. It expects as a parameter either one, two, or off.

Citations

Citations are processed by the `DSMCIT` and `DSMECIT` macros, which turn on and off highlighting around the text of the citation.

Examples

Examples are processed by the `DSMXMP` and `DSMEXMP` macros, which enclose the example in a keep. The `DEPTH` attribute value, if specified, is passed directly to the `.SP` control word.

Figures

Figures are processed by the `DSMFIG` and `DSMEFIG` macros; the figure caption and description tags are mapped to the `DSMFCAP` and `DSMFDESC` macros, respectively, by the `DSMFIG` macro. Figures are enclosed in floats unless the `PLACE` attribute, processed by the `DSM@PLCE` macro, specifies inline; in this case, the figure is enclosed in a keep.

The `ID` attribute is processed by the `DSM@IDS` macro.

If the `FRAME` attribute is `box`, the figure is enclosed in a column- or page-wide box. The `.HR` [Horizontal Rule] or `.SX` [Split Text] control words are used to produce a horizontal rule or to print a string below the figure if it is a top float, above the figure if it is a bottom float, or above and below the figure if the figure is inline.

The `DSMFCAP` macro assigns the figure number, formats that number along with the caption at the bottom of the figure, and adds the caption to the macro `#FIGLIST` for eventual inclusion in the List of Illustrations. The depth is passed directly to the `.SP` [Space] control word.

Note: Because the figure is enclosed within a float or keep, the page on which it will be placed is not known at the time it is formatted. For this reason, the page reference added to the `#FIGLIST` macro is symbolic; its value is substituted only when the List of Illustrations is formatted and the placement of the figure is known. The `#FIGLIST` macro doesn't actually exist in the `maclib`. It is created dynamically as figure captions are processed.

List of Illustrations

The list of illustrations is produced by the DSMFLIST macro, which begins a new page with a level-1 heading and executes the #FIGLIST macro, which is built by the DSMFCAP macro from captions provided with each figure.

Footnotes

Footnotes are processed by the DSMFTNT and DSMEFTNT macros, which enclose the text in a footnote. The DSM@IDS macro processes the ID attribute, and the DSM#SUPR macro prefixes the text with a serial superscript number.

Headings

The seven levels of headings are processed by the DSMHEAD0 through DSMHEAD6 macros, and the ID attribute is processed by the DSM@IDS macro. The text of the level-0 and level-1 headings is saved in the symbol &@head for use in the running footing.

If the symbol &@head1 exists, level-one headings are prefixed with that symbol and with the value contained in the head-level counter. The symbol &@head1 is set to the string "Appendix" by the appendix tag and is set to the contents of the symbol &@bodyhead1, if it exists, by the :BODY tag.

Each of the heading macros changes the mapping of the paragraph tag to an alternate APF, so that the first paragraph after a heading is formatted slightly differently than other paragraphs.

Highlighted Phrases

The highlighted phrase tags are processed by the DSMHP0 through DSMHP3 macros, which begin fonts named hi0 through hi3. Highlighted phrase end-tags are processed by the DSMEHP macro, which is a .PF [Previous Font] control word. The highlight fonts, which are defined in DSMPROF4, are also used by the head-levels as defined in the profile.

Lists

All lists are processed by the DSMLISTM and DSMELIST macros. The DSM#LTYF macro determines the type of list. List item identifiers for each level of nested list are defined in the profile, DSMPROF4, using the .DV [Define Variable] control word. When lists are nested within other lists, formatting parameters such as the indentation and item number are saved in the symbol array &@nest@l during the inner list.

The list item tag is processed by the DSMLITEM macro, which increments the item counter and prefixes the text that follows with an item identifier. The DSMLITEM macro also changes fonts when required.

The definition term and description tags are processed by the DSMDTERM and DSMDDEF macros. The DSMDTERM macro saves the definition term as the item identifier for the DSMDDEF macro. The definition term and description heading tags are processed by the DSMDTHD and DSMDDHD macros.

The glossary list is processed by the DSMGLIST macro. The glossary term and glossary description tags are processed by the DSMGTERM and DSMGDEF macros.

Indexing

The index tags are processed by the DSMINDX1, DSMINDX2, and DSMINDX3 macros, and the index header tags are processed by the DSMIHD1, DSMIHD2, and DSMIHD3 macros. The ID and REFID attributes are processed by the DSM@IDS and DSM@RIDI macros, respectively.

The current index terms are saved in the symbols &@it1, &@it2, and &@it3. The secondary and tertiary tags copy their higher-level terms into the symbols &#it1 and &#it2. If the REFID attribute is entered, the DSM@RIDI macro overlays these symbols with the terms of the referenced index entry. The PAGeref attribute is processed by DSM@PGRF.

Long Quotations

Long quotations are processed by the DSMLQUOT and DSMELQU macros, which determine any current-level indentation and establish the correct indentation for the long quotation. The indentation is set to its previous level when the long quotation is ended.

Paragraphs

Ordinarily, paragraphs are processed by the DSMPARA macro. For performance considerations, the macro is self-modifying: The .IL control word is retained in the macro only if the amount of indentation is not zero.

The heading macros remap the paragraph tag to alternate APFs to provide slightly different formatting of the first paragraph after a heading. The level-0 and level-1 headings map paragraphs to DSMPARA1, the level-2 through level-4 headings map paragraphs to DSMPARA2, and the level-5 and level-6 headings map paragraphs to DSMPARA5.

The DSMPARA1 and DSMPARA2 macros do not indent the first line of the first paragraph. The DSMPARA5 macro formats the paragraph in-line with the heading and supplies the colon to end the heading.

Row Definitions

Row definitions, as created with the :RDEF tag, are used in building tables. Prior to starting a table, all the row definitions used in the table must be defined with the :RDEF tag. The :RDEF tag is processed by the DSMRDEF macro. This tag is used to describe all the characteristics about a particular row.

Attributes are processed by these macros as follows:

Attribute	Macro
ID	DSM#CKID
HP	DSM@HP

ALIGN	DSM@ALI
CONCAT	DSM@CONC
VALIGN	DSM@VALI
ROTATE	DSM@ROT
MINDEPTH	DSM@MIND
ARRANGE	DSM@ARR
CWIDTHS	DSM@CWID
SHADE	DSM@SD

The identifier specified on the :RDEF tag is used on the REFID attribute tag of the :TABLE, :ROW, :THD and :TFT tags to indicate which definitions to use when starting those objects.

Tables

Tables are processed by the DSMTABLE and DSMETBLE macros. Any row definitions used in the table must be defined with the :RDEF tag before using the :TABLE tag.

The REFID attribute is processed by the the DSM@RFID macro. The ID attribute is processed by the DSM@IDS macro. The SPLIT attribute is processed by the DSM@SPLI macro. The WIDTH attribute is processed by the DSM@WID macro.

The table caption and description tags are mapped to the DSMTCAP and DSMTDESC macros, respectively, by the DSMTABLE macro. The DSMTCAP macro assigns the table number, formats that number along with the caption at the bottom of the table, and adds the caption to the macro #TBLLIST for eventual inclusion in the List of Tables.

Once a table is started, a table header and table footer can be defined to be placed at the top and bottom of the table. Table headers and footers are processed by the DSMTHD, DSMETHD, DSMTFT, and DSMETFT macros, respectively. The REFID attribute is processed by the DSM@RFID macro.

Once a table is started, different rows in the table are started using the :ROW tag. Rows are processed by the DSMROW and DSMEROW macros. The REFID attribute is processed by the DSM@RFID macro. The SPLIT attribute is processed by the DSM@SPLI macro.

Once a row, table header, or table footer is started, different cells in the row, header, or footer are started using the :C tag. Cells are processed by the DSMCELL and DSMECELL macros.

Table notes can be placed within a table, using the :TNOTE tag. Table notes are processed by the DSMTNT and DSMETNT macros.

List of Tables

The list of tables is produced by the DSMTLIST macro, which begins a new page with a level-one heading and executes the #TBLLIST macro, which is built by the DSMTCAP macro from captions provided with each table.

Process-Specific Controls

The process-specific control tags are processed by the DSMPSC and DSMEPSC macros, which enclose the element in a conditional section. Conditional section number 9 is used, leaving sections 1 through 8 available for other uses.

The PROC attribute is processed by the DSM@PROC macro, which instructs the formatter to ignore conditional section number 9 if none of the processes specified is valid.

Quotations

Quotations are processed by the DSMQUOT and DSMEQUOT macros, which surround text with quotation marks. The level of nesting of quotations is determined, and the character to be used is selected for the quotation delimiter. The hexadecimal codes for the quote characters are defined in DSMPROF4.

Title Page

The title page tags are processed by the DSMTTLEP and DSMETTLP macros. The DSMTTLEP macro maps the title, document number, document date, and author tags to the DSMTITLE, DSMDCNUM, DSMDATE, and DSMAUTHR macros, respectively. These macros save the text that follows them in symbols named &@title, &@docnum, &@date, and &@author.

If a title page has been requested with the SYSVAR T option, the DSMETTLP macro calls the DSM#TIPG macro to create the title page, as illustrated in Figure 10 on page 113.

General Processing

Identifiers

Unique identifiers are recognized by the figure, table, footnote, list item heading, and index tags. The ID attribute for these tags is processed by the DSM@IDS macro.

Information relevant to the element is saved when the ID attribute is processed and recalled when the appropriate reference tags are processed by the DSMFGREF, DSMFNREF, DSMLIREF, DSMHDFREF, DSMIREF, and DSMTREF macros. The index tags also recognize the REFID attribute.

Identifiers are saved as they are encountered in the symbol arrays &@xref@f, &@xref@n, &@xref@d, &@xref@h, &@xref@i, and &@xref@t. These arrays are used in creating the ID cross-reference listing.

For each figure identifier, the following information is saved in symbols whose names include the identifier name:

&F1@.... The figure number

&FP@.... The page on which the figure is placed

&FF@.... The name of the source file in which the figure is defined

&FL@.... The array index of the element of **&@xref@f** containing the identifier

&FX@.... An array containing the numbers of pages that reference the identifier.

The **&FL@....** symbol is used only when a forward reference to an ID is encountered: the identifier is added to the array **&@xref@f** so that if it is never defined, the error will appear in the cross-reference. If the identifier is later defined, the symbol **&FL@....** is used to remove the first occurrence of the identifier from **&@xref@f**, ensuring that the identifier appears only once in the cross-reference and in the correct order.

Footnote, heading, list item, index, and table identifiers result in similar symbols, except that the first letter of the symbols is N, H, D, I, and T, respectively. Also, the meaning of the first symbol varies, depending on the type of identifier:

&N1@.... The footnote number

&H1@.... The text of the heading

&D1@.... The list item number

&I1@.... The primary index term

&I2@.... The secondary index term (if any)

&I3@.... The tertiary index term (if any).

&T1@.... The table number

SYSVAR R and W

In CMS and TSO, the IDs used in a document can be saved in a file at the end of the last formatting pass. **SYSVAR W** is used to specify the file name of the ID file to be written. The file type, for CMS, will be DSMREFS. The file contains .SE control word lines defining a set of symbols for each ID used. The symbol names are identical to those described above, except that lowercase letters are used instead of uppercase. For example, the symbol that contains the text of a heading would be named **h1@....** instead of **H1@....**.

A file created with **SYSVAR W** can then be used as input for a subsequent formatting run. This is done by specifying the name of the file on the **SYSVAR R** option of the command. The file named with **SYSVAR R** is imbedded at the beginning of the formatting run. The symbols defined in the file are used to resolve cross-references that cannot otherwise be resolved.

Imbedding the ID file allows forward references to IDs to be resolved more accurately on the first pass. It also permits part of a document to be formatted with resolved references to figures and headings outside of the part being formatted.

Note: **SYSVAR R** and **W** are valid *only* in CMS and TSO.

Running Headings and Footings

The running headings and footings defined during initialization in DSMPROF4 contain several unresolved symbols that are substituted on each page. They are:

- &@sec** document security classification, set by the DSM@SEC macro, which processes the SEC attribute of the :GDOC tag.
- &@stitle** document title (or short title), set by the DSMTITLE macro and by the DSM@STTL macro, which processes the STITLE attribute of the :TITLE tag.
- &@shead** chapter heading (or short heading), set by the DSMHEAD0, DSMHEAD1, DSMABSTR, DSMPREF, DSMTOC, DSNFLIST, and DSMINDEX macros and by the DSM@SHD macro, which processes the STITLE attribute of the :H0 and :H1 tags.

You may want to use these symbols if you redefine the running heading or footing.

Messages and Text Constants

All macros that issue messages do so by calling the macro DSM#M** with the message number as the first parameter. The "***" is replaced with a number that corresponds to the language requested. Other parameters can be passed to the macro, depending on the message.

The DSM#M** macro contains the text of all messages issued by the starter set. It appends the current page number to the message and invokes the .MG control word.

All text constants, such as the word 'Figure,' that are used in the starter set are defined in the DSM#S** macro, where "***" is replaced with a number that corresponds to the language requested.

Imbed Tracing

If a cross-reference listing is requested with the SYSVAR X option, the .IM control word is replaced during initialization with the DSMIM macro.

The DSMIM macro intercepts all imbeds and, except for SCRIPT/VS utility files, issues a message indicating the current pass, page number, and level of imbed file nesting before imbedding the file. During the last formatting pass, this information is saved in the symbol array &@imtrace for use in the cross-reference listing.

Cross-Reference Listing

If a cross-reference listing is requested with the SYSVAR X option, the general document end-tag (EGDOC) calls the macro DSM#XLIST to generate it. If the :EGDOC tag is omitted, the epifile portion of the DSMPROF4 profile calls the DSM#XLIST macro.

The DSM#XLIST macro uses the symbol arrays built by the DSM@IDS, DSMFGREF, DSMFNREF, DSMHDFREF, and DSMID macros to produce the cross-reference listing.

Appendix C. Compatibility with Earlier Releases of SCRIPT/VS

This chapter describes the differences between Release 4.0 and earlier versions of SCRIPT/VS.

Documents you created for processing with Release 3.0 and above don't need to be changed to format with Release 4.0.

Release 4.0 of the Document Composition Facility

- Existing documents created for processing with Release 3.2, 3.1, or 3.0 of SCRIPT/VS will produce comparable results.
- Any user-installation modifications to certain SCRIPT/VS modules will have to be reapplied.
- The default text library has changed from SVTEXT to DSMHYLIB. Release 4.0 uses DSMHYLIB only, it cannot use SVTEXT. DSMHYLIB is used with Release 4.0 only. SVTEXT is used with Release 3.2 only. However, if you have both DSMHYLIB and SVTEXT available at the same time, you can use both Release 4.0 and 3.2.
- Some sort sequences have been changed and new ones have been added for the new languages. Refer to the *Document Composition Facility: SCRIPT/VS Text Programmer's Guide* for information on these sort sequences.
- Any user dictionaries named DAN, FIN, ICE, NOR, POR, or SWE must be renamed.
- When hyphenating in the original 6 non-English languages, the hyphenation may be different than before, because those languages now have an algorithmic hyphenator. There are now 2 algorithmic hyphenators for English (both are used for American English, Canadian English, and United Kingdom English). Unless changed at installation time, the algorithmic hyphenator used in Release 3.2 will be used. If changed at installation time (see the *Document Composition Facility: SCRIPT/VS User's Guide*), the new algorithmic hyphenator will be used for English and can cause different hyphenation points.
- The enhancements made to DCF for Release 4.0 required changing the names of many of the items (profiles, libraries, dictionaries, modules, EXECs, loaders, and generators) shipped with DCF. The items with their default names for Release 3.2 and their new names for Release 4.0 are listed in Table 11 on page 154.

Description	Release 3.2 Name	Release 4.0 Name
Starter Set Profile	DSMPROF3	DSMPROF4
Bar Code Profile	DSMBPROF	No change
SMFF Profile	DSMFPROF	No change
Schedule Profile	DSMSPROF	No change
Transparency Profile	DSMTPROF	No change
Memo Profile	DSMMPROF	No change
Macro Library	DSMGML3	DSMGML4
Value of macro library symbol	DSMMAC32	DSMMAC40
CMS Dictionaries	SVTEXT	DSMHYLIB
Note: For more information about the following items, see the appropriate Program Directory.		
CMS Shared Segment Module	DSMSEG3	DSMSEG4
CMS Shared Segment Module		DSMSEG4X
CMS Shared Segment Module		DSMSEG4B
CMS Disk Resident Module	DSM3	DSM4
TSO Module	DSMTSS30	DSMTSS40
ATMS Module	DSMATS30	No change
DLF/VSE Module	DSMLDS30	No change
DLF/MVS Module	DSMLOS30	No change
FLIP Module in CMS	DSMCMF30	DSMCMF40
FLIP Module in MVS	DSMBOF30	DSMBOF40
FLIP Module in VSE	DSMLDF30	DSMLDF40
Service EXEC (called by DSMSRV40)	DSMSRVXA	DSMSRVX4
Service EXEC	DSMSRV32	DSMSRV40
CMS Disk Resident Generator	DSMGND32	DSMGND40
CMS Shared Segment Generator	DSMGNS32	DSMGNS40
Dictionary Selector and Module Loader	DSMDCT32	DSMDCT40
Manual Installation EXEC	DSMINS32	DSMINS40
SMFF Shared Segment Generator	DSMGSF32	DSMGNS40
CMS EXEC for creating and modifying user dictionaries	DSMDMP3	DSMDMP4
Dictionary TEXTLIB Generator	DSMTXT32	DSMTXT40

Table 11. Items with Name Changes for DCF Release 4.0

Note: Since the Office Document Feature is not changing for Release 4.0, the names of its profile (DSMOPROF) and installation exec (DSMINSOD) will remain the same as for Release 3.2.

- In Release 3.2, if a word is found in an addenda or user dictionary with no hyphenation points, and that word is searched for hyphenation, DCF bypasses the entry and looks for more variations of the word in the dictionaries. For Release 4.0, if a word is found with no hyphenation points, no other dictionaries are searched. If .HY ALG was specified and .HY ALT was not specified, the word is sent to the algorithmic hyphenator.
- The following are the codepages used for example fonts for the starter set. These codepages are used everywhere that T1D0BASE was used for Release 3.2:

Language	Codepage
Danish	T1DDBASE
Dutch	T1DNBASE
English American	T1D0BASE
English Canadian	T1DNBASE
English U.K.	T1DUBASE
French Canadian	T1DNBASE
Finnish	T1DEBASE
French	T1DFBASE
German	T1DABASE
Icelandic	T1DDBASE
Italian	T1DIBASE
Norwegian	T1DDBASE
Portuguese	T1DNBASE
Spanish	T1DSBASE
Swedish	T1DEBASE

Appendix D. Related Publications and Products

Related Products

The following products are related to DCF. For information about ordering any of these products, contact your local IBM Branch Office.

- **MARKUP:** A PC-based editor that helps you create GML documents. Refer to *MARKUP User's Guide and Tutorial*, which you can order with the MARKUP product, 6476161.
- **Publishing Systems BookMaster:** A host-based application that runs under control of DCF and is designed for high-volume in-house publishing applications. This application contains a superset of the GML Starter Set.
- **Publishing Systems BookManager BUILD and BookManager READ:** Host-based IBM licensed programs that let you create and use online books and documents at your terminal in a VM/CMS system.
- **OfficeVision:** OfficeVision provides an integrated electronic office that delivers a broad range of office functions for business communications with the help of an integrated set of services.
- **Standard Generalized Mark-up Language (SGML) TextWrite OS/2 Edition:** TextWrite is a software program that writers can use to create and modify SGML-compliant documents.
- **TextTagger:** TextTagger is a software program that analyzes electronic documents and inserts tagging to comply with the Department of Defense's Computer-Aided Acquisition and Logistics Support (CALS) initiative.
- **SGML Translator DCF Edition:** A program that parses, validates, and translates SGML documents so they can be formatted and printed by DCF.
- **Print Services Facility (PSF):** A licensed program that combines print data with resources to manage and control data transmitted to IBM page printers.

Optional Features

For information about ordering these optional features, contact your local IBM Branch Office:

- **SCRIPT Mathematical Formula Formatter (SMFF):** An optional feature of DCF that can be ordered separately. SMFF makes it possible to construct and display mathematical and scientific formulas on page printers.
- **Office Document Feature (ODF):** An optional feature of DCF that can be ordered separately. ODF allows documents that have been created with an office system to be printed on DCF-supported printers.

Publication Library Guide for the Document Composition Facility

The following table lists the Document Composition Facility publications by number as they relate to user tasks. "DCF Publications" on page 159 lists the titles and the order numbers that correspond to the numbers listed in the table.

Number	User Tasks	Typical Audience	Brief Description
(1) (2) (3) (20)	Planning and introducing DCF/DLF	Users, system planners	Provide a general overview of text processing, library facility, and available books.
(3) (4) (5) (12) (16) (20)	Formatting documents (using the GML starter set)	Novice to experienced end users	Provide an introduction to the Generalized Markup Language (GML) starter set and describes the GML starter set tags and SCRIPT/VS messages.
(6)	Creating bar codes with DCF/GML	Experienced end users	Provides information about using GML to create bar codes.
(9) (10) (11) (12) (17) (18) (19) (21)	Formatting documents (using SCRIPT/VS control words)	Knowledgeable to experienced end users	Describe the function and use of all SCRIPT/VS control words, macros, diagnostic aids, and the formatting features and messages.
(14) (15)	Converting RFTDCA for SCRIPT/VS formatting	Novice to experienced RFTDCA users	Describe the function and use of the optional Office Document Feature, including diagnostic aids and messages.
(4) (5) (7) (9) (10) (11) (19)	Modifying GML starter set ⁹	Document administrator and text programmer ¹⁰	Contain material about GML starter set tags, SCRIPT/VS control words, and GML starter set modifications.
(4) (5) (7) (8) (9) (10) (11) (16) (19)	Creating GML application processing functions	Document administrator and text programmer ¹⁰	Provide information about designing your own GML and about GML concepts, GML starter set tags, SCRIPT/VS control words, and usage guidelines.
(10) (11) (12) (13) (19) (24) (25) (26) (27) (28) (29)	Installing, modifying, and maintaining DCF	System programmer	Give information on error isolation, program tailoring, and use of SCRIPT/VS.
(22)	Creating mathematical formulas with SMFF	Experienced end users	Describes the function and use of the SCRIPT Mathematical Formula Formatter (SMFF), including .EQ control word and messages.
(23)	Creating memos, transparencies, and schedules with GML applications	Novice to experienced end users	Describes the use of the memo, transparencies, and schedule applications, including messages.

⁹ Central Programming Service support and maintenance are provided *only* on the unmodified GML applications shipped with DCF. If you modify any of these GML applications shipped with DCF, it is recommended that you also maintain an *unmodified* copy for diagnostic purposes.

¹⁰ The document administrator is responsible for defining markup conventions and procedures for an organization. The text programmer implements application processing functions (APFs) that provide the processing specified by the document administrator.

DCF Publications

Number Titles and Order Numbers

- (1) *About DCF*, G520-6362.
- (2) *Document Composition Facility and Document Library Facility General Information*, GH20-9158.
- (3) *Document Composition Facility: Introduction to Generalized Markup Language*, G544-3192.
- (4) *Document Composition Facility: Generalized Markup Language Starter Set User's Guide*, SH20-9186.
- (5) *Document Composition Facility: Generalized Markup Language Starter Set Reference*, SH20-9187.
- (6) *Document Composition Facility: Bar Code User's Guide*, S544-3115.
- (7) *Document Composition Facility: Generalized Markup Language Starter Set Implementation Guide*, SH35-0050.
- (9) *Document Composition Facility: SCRIPT/VS User's Guide*, S544-3191.
- (10) *Document Composition Facility: SCRIPT/VS Text Programmer's Guide*, SH35-0069.
- (11) *Document Composition Facility: SCRIPT/VS Language Reference*, SH35-0070.
- (12) *Document Composition Facility: SCRIPT/VS Messages*, SH35-0048.
- (13) *Document Composition Facility: Diagnosis Guide and Reference*, LH40-0209.
- (14) *Document Composition Facility: Office Document Feature User's Guide*, G544-3129.
- (15) *Document Composition Facility: Office Document Feature Reference*, S544-3130.
- (16) *Using the Document Composition Facility*, SR21-0515 (Training Course 32291).
- (17) *Using DCF with the 4250 Printer*, SR20-8486 (Training Course 32908).
- (18) *Using DCF with Page Printers*, SR21-1211 (Training Course 32243).
- (19) *Document Composition Facility—Release 3 (SCRIPT/VS) for Document Administrators and Text Programmers*, SR20-7525 (Training Course).
- (20) *Document Composition Facility (SCRIPT/VS) Student Text*, SC20-1894 (Training Course).
- (21) *Document Composition Facility: TSO Enhancements Update Guide*, G544-3345.
- (22) *Document Composition Facility: SCRIPT Mathematical Formula Formatter User's Guide*, S544-3306.
- (23) *Document Composition Facility: Generalized Markup Language (GML) Applications User's Guide*, G544-3305.
- (24) *Document Composition Facility: MVS Program Directory*, G544-3669.
- (25) *Program Directory for use with DCF and SMFF for VM*, G544-3670.
- (26) *Document Composition Facility: VSE Program Directory*, G544-3671.
- (27) *Document Composition Facility: ODF Program Directory for MVS*, G544-3687.
- (28) *Document Composition Facility: ODF Program Directory for VM*, G544-3686.
- (29) *Document Composition Facility: SMFF Program Directory for MVS*, G544-3685.

The following are also available:

- *Document Composition Facility: GML Starter Set Quick Reference*, SX26-3723.
- *Document Composition Facility: SCRIPT/VS Text Programmer's Quick Reference*, SX26-3719.
- *Document Composition Facility Post-Processor Examples*, S544-3484.
- *Document Composition Facility*, SH35-0086 (binder).

Note: The DCF publications will be available at the end of the second quarter of 1991 as displayable BookManager built BOOKs and as source files on a CD-ROM, SK25-1980.

Related Publications

You should use the following publications to evaluate the use of DCF in different operating environments:

- *IBM Virtual Machine/System Product: Introduction*, GC19-6200.

This publication contains an introduction to CMS (Conversational Monitor System), which is one of the interactive systems that SCRIPT/VS operates with.

- *IBM Virtual Machine/System Product: Terminal User's Guide*, GC19-6206.

This publication describes the various terminal types supported by VM/SP for those who plan to use VM/SP in their operations.

- *OS/VS2 TSO Terminal User's Guide*, GC28-0645.

This publication gives detailed user information about OS/VS2 TSO (Time Sharing Option), which is one of the interactive systems that SCRIPT/VS operates with. It describes the TSO EDIT facility and related facilities for text entry, text editing, and data set management.

- *A Guide to IBM's Advanced Function Printing*, G544-3095.

This publication describes the use of a licensed program (PSF, DCF, GML, OGL, GDDM, and PMF) and the use of a subset of a licensed program in conjunction with the IBM Advanced Function Printing (AFP) printers, including the IBM 3800 Printing Subsystem Models 3 and 6 and the IBM 3820 Page Printer.

- *Advanced Function Printing Software: General Information*, G544-3415.

This publication defines Advanced Function Printing (AFP), describes the features and functions of the AFP licensed programs, and shows how the programs work together. It is intended for the people in your organization who will plan for, install, use, and maintain IBM's AFP software products. It also contains a list of the AFP publications.

If you install DLF in the MVS environment, you need a copy of *OS/VS2 Access Method Services*, GC26-3841.

If you install DLF in the VSE environment, you need a copy of *VSE/VSAM Access Method Services: User's Guide and Reference*, SC24-5144.

Glossary

Source Identifiers

This publication includes terms and definitions from:

- The *American National Dictionary for Information Processing Systems*, copyright 1982 by the Computer and Business Equipment Manufacturers Association (CBEMA). Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *ISO Vocabulary—Information Processing* and the *ISO Vocabulary—Office Machines*, developed by the International Organization for Standardization, Technical Committee 97, Subcommittee 1. Definitions of published segments of the vocabularies are identified by the symbol (I) after the definition; definitions from draft international standards, draft proposals, and working papers in development by the ISO/TC97/SC1 vocabulary subcommittee are identified by the symbol (T) after the definition, indicating final agreement has not yet been reached among participating members.

References

The following cross-references are used in this glossary:

Deprecated term for

Indicates that the term should not be used (because it is obsolete, misleading, ambiguous, or jargonistic) and refers to the preferred term. For a deprecated term, the commentary contains only this reference; the deprecated term is not defined.

Synonymous with

Appears in the commentary of a preferred term and identifies less desirable or less specific terms that have the same meaning. The commentaries of the less desirable or less specific terms refer back to the preferred term with the *Synonym for* reference words.

Synonym for

Appears in the commentary of a less desirable or less specific term and identifies the preferred term that has the same meaning.

Contrast with

Refers to a term that has an opposite or substantively different meaning.

See Refers to a multiple-word term in which this term appears.

See also

Refers to related terms that have similar (but not synonymous) meanings.

A

Adobe Document Structuring Conventions. A standard subset of conventions that allow PostScript page descriptions to be accepted as input by many programs and resource managers.

Adobe Font Metrics (AFM) files. Files containing PostScript font information that SCRIPT/VS uses to format documents for PostScript devices.

advanced function printing (AFP). The ability of licensed programs to use the all-points-addressable concept to print text and images on a printer.

alignment. The horizontal placement of text in a column or cell.

all-points addressability. The capability to address, reference, and position text, overlays, and images at any defined point on the print-

able area of the paper. See *page device* and contrast with *line device*.

all-points addressable (APA). In computer printing, pertaining to the ability to address and print or not print each picture element (pel) on a page.

alphameric. Synonym for alphanumeric.

alphanumeric. Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. (A) Synonymous with alphameric.

alphanumeric character set. A character set that contains both letters and digits and may contain control characters and special characters. (T) Synonymous with alphameric.

alphanumeric character subset. A character subset that contains both letters and digits and may contain control characters, special characters, and the space character. (I) (A) Synonymous with alphameric.

alphanumeric string. A sequence of characters consisting of the letters a through z and the numerals 0 through 9.

ampersand. The & character.

When an ampersand begins a character string, SCRIPT/VS assumes the character string is a symbol name. If the symbol name is defined, SCRIPT/VS replaces the symbol with its value (unless symbol substitution is off).

APF. Application processing function.

application processing function (APF). In GML processing, the processing that is performed when a document element or attribute is recognized. In SCRIPT/VS, an APF is implemented as a sequence of control words, possibly intermixed with text and symbols, in one of three forms: macro definition, value of a symbol, or imbedded file.

application program interface (API). The formally-defined programming language interface which is between an IBM system control program or a licensed program and the user of the program.

ascender. (1) In a font, the distance from the baseline to the top of the character. See *maximum ascender*. (2) The part of a lower-case letter that rises above the body of the letter. Letters with ascenders are b, d, f, h, k, l, and t.

ASCII. American National Standard Code for Information Interchange. The standard code, using a coded character set consisting of 7-bit coded characters (8-bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A) Contrast with *EBCDIC*.

Note: IBM has defined an extension to ASCII code (characters 128–255) that uses all 8 bits. DCF uses this 8-bit extension.

ATMS-II. Advanced Text Management System.

ATMS-III. Advanced Text Management System.

attribute. A characteristic of a document (or document element) other than its type or content. For example, the security level of a document or the depth of a figure.

attribute label. In GML markup, a name of an attribute that is entered in the source document when specifying the attribute's value.

B

back matter. In a book, those sections such as glossary or an index that are placed after the main chapters or sections.

balancing. In multicolumn formatting, the process of making column depths on a page approximately equal by redistributing the text in the columns. See also *vertical justification*.

bar code. A code representing characters by sets of parallel bars of varying thickness and separation that are read optically by transverse scanning.

baseline. In a font, the imaginary line on which the bottom of each successive character is aligned.

basic document element. In a general document, one of a group of elements that occurs frequently; for example: note, paragraph, and definition list.

batch environment. An environment in which noninteractive programs are executed.

binding edge. The edge of a page to be bound, stapled, or drilled. Defined with the BIND option of the SCRIPT command.

body. Of a printed page, the portion between the top and bottom margins that contains the text.

boldface. A heavy-faced type. Also, printing in this type.

bottom margin. On a page, the space between the body or the running footing, if any, and the bottom edge of the page.

%%BoundingBox: comment. An Adobe Document Structuring Conventions comment that contains integers describing the coordinates of the lower-left and upper-right corners of the bounding box in the default user coordinate system. The %%BoundingBox: comment coordinates are used by SCRIPT/VS to place the PostScript image in the space reserved within a DCF document.

bounding box coordinates. The coordinates of the lower-left and upper-right corners of an imaginary box surrounding an image in the default user coordinate system. The bounding box coordinates are used by SCRIPT/VS to place a PostScript image in the space reserved within a DCF document.

break. An interruption in the formatting of input lines so that the next input line is printed on a new output line.

bug. An error in a program or in a document markup.

C

call. To transfer control to a procedure, program, routine, or subroutine.

camera-ready copy. Copy which is ready for photographic typesetting.

caps. Capital letters. See also *initial caps*.

caption. Text accompanying and describing an illustration.

case-sensitive. The relevance of a group of letters is uppercase or lowercase. ABC is different from Abc, which is different from ABc.

CDPF. Composed Document Printing Facility.

cell. A single unit within a table, in which text or other expressions may appear. A cell is always rectangular and is usually bounded by horizontal and vertical rules.

centimeter (cm). A measurement equal to one hundredth of a meter; 0.39 inch.

character. A member of a set of elements that is used for the representation, organization, or control of data. Characters may be letters, digits, punctuation marks, or other symbols.

character arrangement table. An array of data that translates input data into printable characters and identifies associated character sets and graphic character modification modules.

character printer. (1) The identification of characters by automatic means. (I) (A) (2) The identification of geographic, phonic, or other characters by various means, including magnetic, optical, or mechanical means. (A)

character set. A finite set of different characters that is complete for a given purpose. For example, in printing, the characters that constitute a font.

character space. The horizontal space of a character. This size depends on the character font and the device on which the character is printed.

character spacing. The space between characters in a word.

cicero. In the Didot point system, a unit of 0.1776 inch (4.512 millimeters) used in measuring typographical material.

CMS. Conversational monitor system—an interactive processor that operates within VM/370.

code page. A font library member that gives the associated code points and character identifiers.

code point. A 1-byte code representing one of 256 potential characters.

coded font. (1) The combination of a code page and a font library. (2) A font library member that is fully described in terms of typeface, point-size, weight, width, and attribute.

color separation. The process of making separate masters of a document for color printing.

column. A vertical arrangement of characters or other expressions.

column balancing. The process of redistributing lines of text among a set of columns so

that the amount of text in each column is as equal as possible.

column width. The width of each text column on a page. Specified with the .CW [Column Width] control word. (In multicolumn formatting, all columns on a page usually have the same width.)

command. A request from a terminal or a request specified in a batch processing job for the performance of an operation or the execution of a particular program. For example, a request given at a terminal for SCRIPT/VS to format a document or for an editor to edit a line of text.

comment. A control word line that is ignored by SCRIPT/VS. Such lines begin with either .* or .cm.

composed text. Text that has been formatted and that contains control information to direct the presentation of the text on page printers.

composite. The act or result of formatting a document.

composite rotation. The total amount of rotation done by the printer to place text in the correct orientation on the page.

composition. The act or result of formatting a document.

compositor. A person or program that composes text.

concatenation. An operation that forms an output line containing as many words as the column width allows, by placing the first words from an input line after the last words from the preceding input line. When words from an input line would reach beyond the right margin and hyphenation cannot be performed, they are placed at the beginning of the next output line, and so on.

control word. An instruction within a document that identifies its parts or indicates to SCRIPT/VS how to format the document. See also *macro*.

control word line. An input line that contains at least one control word.

control word statement. A control word and its parameters.

Conversational Monitor System (CMS). A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities,

and operates only under control of the VM/370 VM control program.

copy group. A portion of a form definition that defines a set of modifications that can be used when printing a page.

current left margin. The left limit of a column that is in effect for formatting. Each column's left margin is specified with the .CD [Column Definition] control word. However, the current left margin (that is, the left boundary for an output line) might vary to the right of the column's left margin when indentation is changed with the .IN [Indent], .UN [Undent], .IL [Indent Line], and .OF [Offset] control words.

current line. The line in a source document at which a computer program (such as an editor or a formatter) is positioned for processing.

D

debug. (1) To detect, diagnose, and eliminate errors in computer programs and SCRIPT/VS documents. (T) (2) Synonymous with checkout, troubleshoot.

default value. A value assumed by a computer program when a control word, command, or control statement with no parameters is processed. In GML processing, the value assumed for an attribute when none has been specified.

deferred control words. SCRIPT/VS control words that are processed after the text has been placed on the page.

descender. (1) In a font, the distance from the baseline to the bottom of the character. See *maximum descender*. (2) The part of a letter that falls below the body of the letter. Letters with descenders are g, j, p, q, y, and Q.

destination. (1) Any point or location, such as a node, station, or particular terminal, to which information is to be sent. (2) An external logical unit LU or application program to which messages or other data are directed.

dictionary. A collection of *word stems* that is used with the spelling verification and automatic hyphenation functions.

Didot point system. A standard printer's measurement system on which type sizes are based. A Didot point is 0.0148 inch (0.376

millimeter). There are 12 Didot points to a cicero. See also *cicero* and *point*.

document. A publication or other written material. See also *output document* and *source document*.

document administrator. A person who is responsible for defining markup conventions and procedures for an organization.

document conversion processor. A computer program that processes a machine-readable document that includes formatting controls written in one formatter language, to produce a machine-readable document that includes formatting controls appropriate for another formatter language.

document library. A set of VSAM data sets, accessible in a batch environment, that contain documents and related files.

dot leader. A set of periods that fills in the space between two pieces of split text such as a chapter title and its page number in a table of contents.

download. To transfer data from a processing unit to an attached device such as a micro-computer for processing. Contrast with *upload*.

duplex. A mode of copying or printing on both sides of a sheet.

E

EBCDIC. Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

edit. To add, change, delete, or rearrange data and to perform operations such as code conversion and zero suppression.

editor. (1) See *linkage editor*. (2) See *editor program*.

editor program. (1) A computer program designed to perform such functions as rearrangement, modification, and deletion of data in accordance with prescribed rules. (2) Contrast with *linkage editor*.

eject. In formatting, a skip to the next column or page.

element. Any part of a document: a single character, a word, or a sentence. Also refers to any part of a document you can identify with a GML tag (tagged element), such as a paragraph, figure, or heading.

em. A unit of measure equal to the width or the height of the character "m" in a particular font.

en. A unit of measure usually equal to one-half the width of an em. For many typefaces, the average width of lower case characters tends to be equal to the width of an en.

enabled. Used in reference to a tag, it means that the tag is mapped to its appropriate APF.

Encapsulated PostScript. Any file containing the PostScript Page Description Language that conforms to Adobe 2.0 Document Structuring Conventions and follows conventions defined by Adobe Systems to allow the file to be included by other applications. Encapsulated PostScript can be included in a DCF document by means of the .PO [PostScript] control word.

epifile. The portion of a profile (after a .EF control word) that is processed *after* the main document has been processed.

escapement. (1) Movement of one character space between the paper carrier and typing or printing position, parallel with the typing or printing line. (2) The unit of vertical or horizontal movement that is built into a device. For the 4250, that value is 1/600th of an inch; for the 3800 Model 3 and the 3820 Page Printer, that value is 1/240th of an inch; and for PostScript devices, that value is 1/72000th of an inch.

exposure. The amount of risk associated with a schedule item or items.

extended symbol processing. The processing of a symbol whose value causes the rest of the line to be stacked and later processed as a new input line.

F

factor. A dimensionless scalar value used to form a product with another value. Factors can also be expressed as percentages.

FBA. Fixed-block-architecture.

figure space. (1) A unit of measure equal to the width of the "en" space in a particular font. (2) In the Document Composition Facility, the width of the figure (0).

fill character. A character used to occupy a space; for example, blanks used to fill up the space left by tabbing.

float. (1) A keep (group of input lines kept together) whose location in the source file can vary from its location in the printed document. (2) Of a keep, to be formatted in a location different from its location in the source file.

flush. Having no indentation.

fold. (1) To translate the lowercase characters of a character string into uppercase. (2) To place that portion of a line that does not fit within a column on the next output line.

folio. Page number.

font. A font library member that contains characters that must be used in conjunction with a code page font library member.

font object. A member of a font library. In CMS, a file whose filetype matches the name of the font library. In MVS, a member of a partitioned data set (PDS).

font set. The set of fonts to be used in formatting a source document.

footer. Text that appears at the bottom of every page of a document, for example, a page number.

footing. Words located at the bottom of the text area. See also *running footing*.

footnote. A note of reference, explanation, or comment, placed below the text of a column or page, but within the body of the page above the running footing.

foreground. The environment in which interactive programs are executed. Interactive processors reside in the foreground.

format. (1) The shape, size, and general makeup of a printed document. (2) To prepare a document for printing in a specified format.

formatter. (1) A computer program that prepares a source document for printing. (2) That part of SCRIPT/VS that formats input lines for a particular type of logical device.

formatting mode. In document formatting, the state in which input lines are concatenated and the resulting output lines are justified.

FORMDEF. Synonym for form definition.

form definition (FORMDEF). A resource object that defines the characteristics of the

form which include: overlays to be used, text suppression, position of page data on the form, and number and modifications of a page. Synonymous with FORMDEF.

front matter. In a book, those sections (such as preface, abstract, table of contents, list of illustrations) that are placed before the main chapters or sections.

G

general document. A type of document whose description can apply to a variety of documents, from memoranda to technical manuals. It can be used as a catch-all category for documents that do not conform to any other type description.

Generalized Markup Language (GML). A language that can be used to identify the parts of a source document without respect to a particular processing system.

GML delimiter. A special character that denotes the start of GML markup. In the starter set, it is initially a colon (:).

GML end tag delimiter. A special character that denotes the end of GML markup. In the starter set, it is initially a period (.).

GML interpretation. Recognizing the start or end of an element (or an attribute label), associating it with an APF, and executing the APF. In SCRIPT/VS, interpretation is performed jointly by SCRIPT/VS and by APFs.

graphic character modification (GCM) module. Modules that contain scan patterns of IBM-supplied character sets, user-defined character sets, or both without respect to particular processing.

graphical data display manager (GDDM). An IBM licensed program that creates page segments.

gutter. In multicolumn formatting, the space between columns.

H

hanging indentation. The indentation of all lines of a block of text following the first line, which is not indented the same number of spaces. Specified with the .OF [Offset] or .UN [Undent] control word.

heading. Words located at the beginning of a chapter or section or at the top of a page,

above the first line of text on the page. See also *head-level* and *running heading*.

heading segment. An element that begins with a heading, followed by basic document elements and lower-level heading segments.

head-level. The typeface and character size associated with the words standing at the beginning of a chapter or chapter topic.

hexadecimal. Pertaining to a numbering system based on 16, using the sixteen digits 0, 1, . . . 9, A, B, C, D, E, and F. For example, hexadecimal 1B equals decimal 27. See also *EBCDIC*.

highlighting. Emphasis associated with a document element. In formatting, highlighting is usually expressed by changing a font, overstriking, underscoring, or capitalizing the highlighted element.

horizontal justification. The redistribution of horizontal white space at the end of a line of text to the spaces between the words and letters of the line in order to exactly fill the width of the column with the text.

I

IEBIMAGE. A utility program that creates and maintains various 3800 Printing Subsystem Model 1 modules (for example, character arrangement table and graphic character modification (modules) and stores them in SYS1.IMAGELIB.

image. (1) A likeness or imitation of an object, such as a picture or logo. (2) A PostScript file that can contain any combination of images and text.

impact printer. A printer, in which printing is the result of mechanical impacts. (I) (A)

implied paragraph structure. An element that begins with an implied paragraph; that is, one for which you do not specifically enter a paragraph tag. The existence of the paragraph is understood from the existence of the implied paragraph structure, for example, as with notes, figure captions, and lists.

indent. To set typographical material to the right of the left margin, while still retaining the original (fixed) margin settings.

indentation. The action of indenting. The condition of being indented. The blank space produced by indenting. Specified with the .IN [Indent], .IR [Indent Right], .UN [Undent],

.OF [Offset], and .IL [Indent Line] control words. See also *hanging indentation*.

initial value. A value assumed by SCRIPT/VS for a formatting function until the value is explicitly changed with a control word. The *initial value* is assumed even before the control word is encountered, whereas the *default value* is assumed when the control word is issued without parameters. See also *default value*.

initialize. (1) In programming languages, to give a value to a data object at the beginning of its lifetime. (2) To set counters, switches, addresses, or contents of storage to zero or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

inline space. The amount of horizontal white space that usually occurs between words in a line.

input device. Synonym for input unit.

input line. A line, as entered into a source file, to be processed by a formatter.

input unit. A device in a data processing system by means of which data can be entered into the system. (I) (A) Synonymous with input device.

interactive. Pertaining to an application in which entries call forth a response from a system or program, as in an inquiry system. An interactive system might also be conversational, implying a continuous dialog between user and system. Interactive systems are usually communicated with via terminals, and they respond to commands. See also *foreground*.

interactive environment. The environment in which an interactive processor operates.

Interactive System Productivity Facility (ISPF). A dialog manager for interactive applications that provides control and services to allow processing of the dialogs in different host environments.

intercharacter space. Extra horizontal white space inserted *between* characters of a word. This space is in addition to the space included as part of the characters by the designer of the font.

interword space. See *word space*.

ISPF. Interactive System Productivity Facility.

italic. A type style with characters that slant to the right.

J

job control language (JCL). A control language used to identify a job or describe its requirements to the operating system.

job control statement (JCS). A statement in a job that is used in identifying the job or describing its requirements to the operating system.

justification. See *horizontal justification* and *vertical justification*.

justify. To control the printing positions of characters on a page so that the left-hand and right-hand margins of the printing are regular. (I) (A) See *left-justify* and *right-justify*

K

kanji. The nonphonetic Japanese writing system. In a font representing kanji characters, each character is represented by a double-byte code. Contrast with katakana.

katakana. A character set consisting of symbols used in one of the two common Japanese phonetic alphabets. Each character is represented by one byte. Contrast with kanji.

keep. In a source document, a collection of lines of text to be printed in the same column. When the vertical space remaining in the current column is insufficient for the block of text, the text is printed in the next column. In the case of single-column format, the next column is on the next page.

L

layout. The arrangement of matter to be printed. See also *format*.

leader. (1) Dots or hyphens (as in a table of contents) used to lead the eye horizontally. (2) The divider between text and footnotes on a page, usually a short line of dashes.

left-hand page. The page on the left when a book is opened; usually even-numbered.

left-justify. To control the printing positions of characters on a page so that the left-hand margin of the printing is regular. (I) (A)

legend. An explanatory list of the symbols, lines, and other components of a schedule.

ligature. A single character (piece of type or font raster) that represents two or more input characters: ff and ffi are examples of characters that may be represented by (printed as) a ligature.

line device. Any of a class of printers that accept one line of text from the host system at a time. SCRIPT/VS supports such line devices as the 1403 Printer and 3800 Model 1.

line printer. (1) A device that prints a line of characters as a unit. (I) (A) (2) Contrast with character printer, page printer.

line space. The vertical distance between the baseline of the current line and the baseline of the preceding line.

line spacing. See *line space*.

logical output device. The combination of a physical output device and such logical variables as page size and number of lines per vertical inch (for line devices). A specification of 1403W6 is an example of a logical output device.

logical page. Synonym for page.

lowercase. Pertaining to small letters as distinguished from capitals, for example: a, b, g rather than A, B, G.

M

machine-readable. Pertaining to data a machine can acquire or interpret (read) from a storage device, or a data medium, or other source.

maclib. See *macro library*

MACLIB library. A library that contains macros, copy files, or source program statements for use under CMS.

macro. See *macro instruction*

macro instruction. (1) An instruction that when executed causes the execution of a pre-defined sequence of instructions in the same source language. In SCRIPT/VS, a macro is a sequence of one or more control words, symbols, and input lines. A macro's definition can be recursive. (2) Synonymous with macrostatement.

macro library. A library of macro definitions used during macro expansion. The form the

library takes will vary by environment, being a MACLIB in CMS, a PDS in TSO, and so on.

macrostatement. Synonym for macro instruction.

macro substitution. During formatting, the substitution of control words, symbols, and text for a macro.

map. To associate a tag with an APF, using the AA [Associate APF] control word.

margin. (1) The space above, below, and on either side of the body of a page. (2) The left or right limit of a column.

mark up. (1) To determine the markup for a document. (2) To insert markup into a source document.

markup. Information added to a document that enables a person or system to process it. Markup information can describe the document's characteristics, or it can specify the actual processing to be performed. In SCRIPT/VS, markup consists of GML tags, attribute labels and values, and control words.

markup-content separator. A delimiter used in GML markup that indicates the end of the markup and the beginning of the text. The default markup content separator for GML is a period (.).

maximum ascender. The maximum height from the baseline of sequential characters to the top mark of the tallest character in a font character set.

maximum descender. The maximum depth from the baseline of sequential characters to the bottom mark of any character in the font character set.

MCS. Markup/content separator.

meter (m). 1.0936 yards; 3.2808 feet; 39.3696 inches.

millimeter (mm). One thousandth of a meter; 0.04 inch.

N

National Language Support. Translation requirements affecting parts of devices and licensed programs; for example, rules for translation of message text, and for conversion of symbols such as the US dollar sign to the UK pound sign.

nonimpact printer. A printer, such as the 3800 Printing Subsystem, in which printing is not the result of mechanical impacts, but is instead produced by another process such as laser beam, ink jet, or electroerosion. The 3800 Printing Subsystem, for example, uses a laser based technology, and the 4250 Printer uses an electroerosion process.

O

object. A sequential collection of control records that represents documents, pages, fonts, and so on.

offset. (1) To indent all lines of a block of text, except the first line. (2) The indentation of all lines of a block of text following the first line.

option. Information entered with a SCRIPT command to control the execution of SCRIPT/VS.

orientation. In the 3800 Printing Subsystem Models 3 and 8, the number of degrees an object is rotated relative to a reference; for example, the orientation of printing on a page, relative to the page coordinates. See also *text orientation*.

output device. Synonym for output unit.

output document. A machine-readable collection of lines of text or images that have been formatted, or otherwise processed, by a document processor. The output document can be printed or it can be filed for future processing.

output line. A line of text produced by a formatter.

output unit. A device in a data processing system by which data can be received from the system. (I) (A) Synonymous with output device.

overlay. A collection of predefined data such as lines, shading, text, boxes, or logos that can be merged with variable data on a page while printing.

P

page. A collection of data that can be printed on a physical sheet of paper. Synonymous with logical page.

PAGEDEF. Page definition.

page definition (PAGEDEF). (1) A resource,

specified in the print data set JCL, that defines the rules for transforming the input to pages and text controls. (2) In the 3800 Print Management Facility, a member of a partitioned data set that contains the formatting instructions for a print data set, although it can be used for any compatible print data set.

page device. A device that prints a formatted page that has graphics and text merged.

page printer. (1) Any of a class of devices that accept composed pages, constructed of composed text and images, and prints one page as a unit. SCRIPT/VS supports such page printers as the 4250 Printer, the 3800 Model 3, and the 3820 Page Printer. (2) Contrast with *character printer*, *line printer*.

page segment. (1) An object that can contain text and images and be included on any addressable point on a page or electronic overlay. It assumes the environment of the object it is included in. (2) A library member that contains the definition of the page segment.

paginate. To number pages.

paragraph unit. An element that has the same structure as a paragraph. In a general document, the paragraph units are: paragraph, note, and paragraph continuation.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (I) (A) The syntax of some SCRIPT/VS control words includes parameters, which establish the properties of a formatting function or a printed page. (2) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted.

part. In a general document, a part is a zero-level heading segment. See also *heading segment*.

patch PSC element. A PSC element that is used temporarily to modify the normal output.

pel. (1) An element of a raster pattern about which a toned area on a photoconductor can appear. See also *raster pattern*. (2) The unit of horizontal measurement for the 3800 Printing Subsystem and the 4250 Printer. On the 3800 Printing Subsystem Model 1, one pel equals approximately 1/180th inch. On the 3800 Model 3 and the 3820 Page Printer, one pel equals approximately 1/240th inch. On

the 4250 Printer, one pel equals approximately 1/600th inch.

pel density. The number of picture elements per inch of linear measurement.

physical output device. A physical device, that stores, prints, or displays data, such as a terminal, a disk file, a line printer, or a nonimpact printer. The 1403 is an example of a physical output device.

pica. A unit of about 1/6 inch used in measuring typographical material. It is similar to a cicero in the Didot point system.

pitch. On a typewriter, the distance between corresponding points of two equal characters that are typed immediately adjacent to one another.

point. (1) A unit of about 1/72 of an inch used in measuring typographical material. There are 12 points to the pica. (2) In the Didot point system, a unit of 0.0147 inch. There are 12 Didot points to the cicero.

PostScript devices. Any of a class of devices that accept the PostScript page description language, such as the IBM 4019 LaserPrinter. When used with DCF, PostScript devices must be configured to accept 8-bit ASCII.

PostScript language. A programming language designed to convey a description of virtually any desired page to a printer. It possesses a wide range of graphic operators that may be combined in any manner.

PostScript image file. Any file containing encapsulated PostScript that is imbedded in a DCF document by means of the .PO [PostScript] control word. PostScript image files can include any combination of images or text.

profile. (1) In SCRIPT/VS processing, a file that is imbedded before the primary file is processed. It can be used to control the formatting of a class of source documents. When processing GML markup, the profile usually contains the mapping from GML to APFs and the symbol settings that define the formatting style. (2) In the DLF library, a collection of information that identifies a batch SCRIPT/VS user (user profile) or a document processor (attribute profile) or that defines certain library parameters (system profile).

proportional spacing. The spacing of characters in a printed line so that each character is

allotted a space proportional to the character's width.

R

ragged right. Pertaining to text that is not right-justified. See also *left-justify*.

ragged left. Pertaining to text that is not left-justified. See also *right-justify*.

reference element. In a general document, an element whose content is a reference to another element that is generated by an APF. There are five: figure reference, footnote reference, heading reference, index entry reference, and list item reference.

required blank. A character that prints as a blank, but does not act as a word separator.

required space. In word processing, a space or blank that must not be removed when adjusting a line or paragraph of text. A character that prints as a blank, but does not act as a word separator. Synonym for required blank.

residual text. The line of text following the markup/content separator of a GML tag.

RFTDCA. Revisable-Form-Text Document Content Architecture specifies how IBM office systems interchange documents that are in revisable form. This architecture defines the structure of the data streams that represent revisable-form-text documents within the office system or network.

right-hand page. The page on the right when a book is opened; usually odd-numbered.

right-justify. To control the positions of characters on a page so that the right-hand margin of the printing is regular. (I) (A)

row. A horizontal arrangement of characters or other expressions on a printed page.

rule. (1) A straight horizontal or vertical line used, for example, to separate or border the parts of a figure or box. (2) A solid or patterned line of any weight, extending horizontally across the row or vertically down the column.

running footing. A footing that is repeated above the bottom margin area on consecutive pages or on consecutive odd-numbered or even-numbered pages in the text area of the page. Synonymous with footer.

running heading. A heading that is repeated below the top margin area on consecutive pages or on consecutive odd-numbered or even-numbered pages in the text area of the page.

S

SCRIPT/VS. The formatter component of the Document Composition Facility. SCRIPT/VS provides capabilities for text formatting and document management, macro processing and symbol substitution, and GML tag recognition and processing.

section. Each part of the output page when an output page has two or more single-column parts with the same or different column-widths, or a single-column part and a multicolumn part, or two or more different multicolumn parts.

segment. An object containing composed text and images, prepared before formatting and included in a document when it is printed.

separation masters. The process of making separate masters of a document for the purpose of special printing (such as multi-part forms or multiple-color printing).

set. Used in reference to a symbol; it implies the .SE [Set Symbol] control word.

set size. The set size of a given typeface determines the number of characters that will fit in a line of a given width when it is printed or set.

shading. Highlighting an area on the page by varying graded density.

showpage. A PostScript command that transmits the current page to the current output device, causing any marks on the page to actually appear.

slip end date. The end date of a schedule or project that has moved or *slipped* to a later time than originally planned.

slip start date. The start date of a schedule or project that has moved or *slipped* to a later time than originally planned.

small caps. Capital letters in the same style as the normal capital letters in a font, but approximately the size of the lowercase letters.

source document. A machine-readable collection of lines of text or images that is used for input to a computer program.

space. A blank area separating words or lines.

space unit. A unit of measure of horizontal or vertical space. In GML markup, the *em* is used when a measure that is relative to the current font size is required. When an absolute measure is required, as in specifying the depth of a figure, recommended space units are inches (*nnI*), millimeters (*nnW*), picas/points (*nnPnn*), or Ciceros/Didot points (*nnCnn*), where *nn* is the number of units. See also *em*, *pica*, *point*, *Cicero*, and *Didot point system*.

starter set. An example of GML support that is provided with the Document Composition Facility. It consists of a document-type description for general documents, a profile, and a library of APFs.

string. A sequence of elements of the same nature, such as characters considered as a whole.

structure. A characteristic of a document (or element) that expresses the type and relationship of the elements of the content. See also *content* and *element*.

structured field. A self-identifying string of bytes, analogous to a logical record. A structured field consists of an introducer, which identifies and characterizes the structured field, and data or parameters.

symbol. A name in a source document that can be replaced with something else. In SCRIPT/VS, a symbol is replaced with a character string. SCRIPT/VS can interpret the character string as a number, a character string, a control word, or another symbol.

symbol substitution. During formatting, the replacement of a symbol with a character string that SCRIPT/VS can interpret as a value (numeric, character string, or control word) or as another symbol.

SYSVAR. An option of the SCRIPT command that permits the user to specify values for symbols. In the starter set, SYSVAR symbol values determine whether certain processing variations will occur, such as heading numbering, duplex formatting, and two-column printing.

T

tab. (1) (noun) A preset point in the typing line of a typewriter-like terminal. A preset point in an output line. (2) (noun) A tab character, *X'05'*. (3) (verb) To advance to a tab for printing or typing.

table. (1) An array of data each item of which can be unambiguously identified by means of one or more arguments. (I) (A) (2) An arrangement of cells in rows and columns.

tag. In GML markup, a name for a type of document or document element that is entered in the source document to identify it. For example, *.p.* is the tag used to identify a paragraph.

terminal. A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a communication channel.

text item. Explicitly marked elements that occur within text, such as within a paragraph unit. In a general document, for example, quotations and phrases are text items.

text orientation. A description of the appearance of text as a combination of print direction and character rotation.

text programmer. One who implements APFs that provide the processing specified by the document administrator. In SCRIPT/VS, this involves writing SCRIPT/VS macros and organizing macro libraries and profile files so that the appropriate composition will be done for each tag.

text line. A line that contains only text.

text variable. A symbol whose final value is to be treated only as text.

time sharing option (TSO). An option on the operating system; for System/370, the option provides interactive time sharing from remote terminals.

token. A string of characters that is treated as a single entity. In SCRIPT/VS, a parameter passed to a macro in one of the local variables *&*1, ... &*n*.

top margin. On a page, the space between the body or running heading and the top edge of the page.

transparency. A master or copy on material that transmits light without diffusion.

translation table. A table that provides replacement characters for characters that cannot be printed or that substitutes characters from a different font. For example, the 256-byte portion of the character arrangement table that translates the user's data code for a character recognizable by the 3800 Printing Subsystem Model 1.

TRC. Table reference character (TRC) in printer SYSOUT data sets, a second control byte, following the carriage control byte, which indicates which font the record is to be printed in. The presence of TRCs is indicated by the JCL parameter DCB=OPTCD=J.

TSO. Time-sharing option.

typeface. All type of a single style. There might be several fonts (different sizes) with the same typeface or style.

typeface family. A collection of fonts of a common typeface that vary in size and style.

typeset. (1) To arrange the type on a page for printing. (2) Pertaining to material that has been set in type.

type posture. A typeface style variation indicating whether a typeface is upright (as in Roman) or slanted to the right (as in italic or cursive).

type size. A measurement in pitch or points of the height and width of a graphic character in a font. For example, the vertical height (point size) of a given typeface, such as 10 point.

type style. The form of characters within a set of the same font, for example: elite, or pica. (T) Attributes such as posture, weight, and width may vary in a type style.

type weight. (1) The degree of boldness of a typeface series, caused by different thicknesses of the strokes that form a graphic character. (2) One of the many attributes of a font, others, for example: being size and typeface.

type width. The horizontal size (set size) of a given typeface. The width may be given in units of measurement, such as set 9 point, or it may be descriptive: ultra condensed, condensed, expanded, and so on.

U

underscore. (1) A line printed under one or more characters. (2) To place a line under one or more characters; to underline.

unformatted mode. (1) In document formatting, the state in which each input line is processed and printed without formatting. Other SCRIPT/VS control words remain in effect and are recognized. (2) In document printing, using the UNFORMAT option, the state in which each input line (control words as well as text) is printed as it exists in the input, in the order in which it is processed. No formatting is done.

unique identifier (ID). In a general document, an attribute whose value serves as a name that can be used to refer to the element. See also *reference element*.

unit space. The minimum amount of additional spacing acceptable for purposes of horizontal justification, as specified by the font designer.

upload. To transfer data from a device such as a microcomputer to a processing unit. Contrast with *download*.

uppercase. Pertaining to capital letters as distinguished from small letters, for example: A, B, G rather than a, b, g.

V

variable. A quantity that can assume any of a given set of values.

variable text. For the .VT [Variable Text] control word, text to be inserted in a formatted document by a postprocessor.

vertical justification. Redistribution of the extra vertical white space at the end of a column between lines of text, so as to make the columns appear to be the same length.

W

widow. A line or word by itself ending a paragraph.

word space. The white space placed between words in a line. This is sometimes referred to as an interword space.

word spacing. The space between words in a line. See also *word space*.

writable character generation module (WCGM). In the 3800 Printing Subsystem, a 64-position portion of character generation storage that holds the scan elements of a single character set. There are two WCGMs in the

basic 3800, and optional added storage provides two more.

X

XPO. See *exposure*

Index

Special Characters

- : (colon), as a GML delimiter 20
- * (period asterisk), using to include comments 28
- (period)
 - using as a markup/content separator 21
 - using as a symbol delimiter 23
- &
 - See ampersand
- &date. symbol 24
- &gml. symbol 24
- &rbl. symbol 24
- &semi. symbol 24
- &time. symbol 24

Numerics

- 3800
 - default fonts 126
- 3800 Printing Subsystem Model 3
 - default file name, file type, file mode for 125
 - default font libraries for 119
 - default page dimensions 130
 - default segment libraries for 123
 - Including overlays 30
 - Including page segments 30
 - Using the SYSVAR option 133
- 3820
 - default fonts 126
- 3820 Page Printer
 - default file name, file type, file mode for 125
 - default font libraries for 119
 - default page dimensions 130
 - default segment libraries for 123
 - Including overlays 30
 - Including page segments 30
 - Using the SYSVAR option 133
- 4250
 - default fonts 126

- 4250 printer
 - default file name, file type, file mode for 125
 - default font libraries for 118
 - default page dimensions 130
 - default segment libraries for 123
 - Including page segments 30
 - Using the SYSVAR option 133

A

- ABSOLUTE parameter ix, 31
- ABSTRACT tag description 37
- adding a line to an address 38
- address line (ALINE tag) 38
- ADDRESS tag description 38
- address, including on title page 38
- ALINE tag description 38
- ampersand
 - symbol 23
 - use of in symbols 23
- APFs
 - See application processing functions
- APPENDIX tag description 39
- application processing functions
 - definition of 4
 - used to detect markup errors 137
- attributes
 - ALIGN (on RDEF tag) 89
 - ARRANGE (on RDEF tag) 91
 - BREAK (on DL tag) 45
 - COLUMN (on TABLE tag) 94
 - COMPACT
 - on DL tag 45
 - on GL tag 60
 - on SL, OL, UL tags 78
 - CONCAT (on RDEF tag) 89
 - CWIDTHS (on RDEF tag) 90
 - DEPTH
 - on FIG tag 49
 - on XMP tag 115
 - FRAME (on FIG tag) 48
 - HEADHI (on DL tag) 45

attributes (*continued*)

HP (on RDEF tag) 89
 ID
 on FIG tag 48
 on FN tag 54
 on H0-H6 tags 62
 on index entry tags 68
 on index header tags 71
 on LI tag 78
 on RDEF tag 88
 on TABLE tag 93
 MINDEPTH (on RDEF tag) 90
 PAGE
 on FIGREF tag 53
 on HDREF tag 65
 on LIREF tag 77
 on TABLE tag 94
 on TREF tag 110
 PG
 on index entry tags 68
 on IREF tag 75
 PLACE (on FIG tag) 49
 PRINT (on index header tags) 71
 PROC (on PSC tag) 85
 REFID
 on FIGREF tag 53
 on FNREF tag 56
 on HDREF tag 65
 on index entry tags 68
 on IREF tag 75
 on LIREF tag 77
 on ROW tag 95
 on TABLE tag 93
 on TFT tag 96
 on THD tag 97
 on TREF tag 110
 ROTATE
 on TABLE tag 94
 SEC (on GDOC tag) 58
 SEE
 on index header tags 71
 on IREF tag 75
 SEED
 on index header tags 71
 on IREF tag 75
 SHADE (on RDEF tag) 92
 SPLIT
 on ROW tag 95
 on TABLE tag 93
 STITLE
 on H0 and H1 tags 62
 on TITLE tag 111
 TERMHI
 on DL tag 45
 on GL tag 60
 TSIZE (on DL tag) 45

attributes (*continued*)

VALIGN (on RDEF tag) 90
 WIDTH
 on FIG tag 49
 on TABLE tag 94
 attributes of a document element 4
 AUTHOR tag description 111
 author's name, including on title page 111

B

back matter section, specifying 41
 BACKM (back matter) tag description 41
 batch
 environment, issuing SCRIPT command
 in 133
 processing 133
 BIND option 118
 blank, symbol for 24
 body section, specifying 42
 BODY tag description 42
 BookManager BUILD and READ 157
 BookMaster 157
 breaking tables 27

C

C (cell) tag description 95
 Cell (C tag) 95
 changes with Release 4.0 ix
 chapter headings, specifying 62
 CHARS (script option) 127
 CHARS option 118
 CICS environment, issuing SCRIPT
 command in 117, 132
 CIT (title citation) tag description 43
 CMS environment, issuing SCRIPT
 command in 130
 COLUMN on TABLE tag 94
 comments
 including in a list 45, 78
 including in a source document 28
 compatibility
 CONCAT on RDEF tag 90
 CONTINUE option 126
 Control Words
 using with GML 26
 Control Words and Macros 26
 Conversational Monitor System 130
 courses
 self-study 159
 training 159

cross-references 19
current date
 including on title page 111
 symbol for 24
current time symbol 24
CWIDTHS on RDEF tag 91

D

date
 including on title page 111
 symbol for 24
DATE tag description 111
DCF publications 159
 library guide 158
DD (definition description) tag
 description 45
DDHD (definition description heading) tag
 description 45
default fonts 126, 127
defining symbols 27
definition
 description (DD tag) 45
 description heading (DDHD tag) 45
 list
 as basic document elements 17
 list tag description 44
 term (DT tag) 45
 term heading (DTHD tag) 44
DEVICE option 124, 131
dictionary
 adding words to 29
DL (definition list) tag description 44
DOCNUM (document number) tag
 description 111
document
 number, including on title page 111
 title, including on title page 111
DSMGML4 maclib 145–152
DSMPROF4 profile 139–144
DT (definition term) tag description 45
DTHD (definition term heading) tag
 description 44
duplex

E

elements of a document
 attributes of 4
 definition of 4
 structural 14
Entering text 22

error handling 126
examples, specifying 115
extended architecture for VM and MVS ix

F

features, optional
 Office Document Feature (ODF) 157
 SCRIPT Mathematical Formula Formatter (SMFF) 157
FIG (figure) tag description 48
FIGCAP (figure caption) tag description 48
FIGDESC (figure description) tag
 description 48
FIGLIST (list of illustrations) tag
 description 52
FIGREF (figure reference) tag description 53
figure
 caption (FIGCAP tag) 48
 description (FIGDESC tag) 48
 list (FIGLIST tag) 52
 reference (FIGREF tag) 53
figure (FIG tag) 48
figures, referencing 53
FILE option 125
file size limitation, removal of ix
flagging changes 28
FLIP enhancement x
FN (footnote) tag description 54
FNREF (footnote reference) tag
 description 56
font character set size limitation, removal
 of ix
font library index program (FLIP) enhance-
 ment x
font requirements
 page printers 126
FONTLIB option 118
fonts, default 126
fonts, page printers 126
fonts, PostScript devices 127
footnote (FN tag) 54
footnote reference (FNREF tag) 56
footnotes, referencing 56
formatting for duplex
FPASSES option 120
front matter (FRONTM tag) 57
FRONTM (front matter) tag description 57

G

GD (glossary definition) tag description 60

- GDOC (general document) tag
 - description 58
- general document (GDOC tag) 58
- Generalized Markup Language
 - advantages of using 5
 - alternative interpretation 5
 - delimiter
 - definition of 20
 - symbol 24
 - symbol instead of colon 24
 - entering 20
 - how vs. interprets it 7
 - introduction to 4
 - markup
 - detecting errors in 137
 - for interim processing 136
 - objectives of 5
 - modifying processing results 33
 - processing text with 5
 - starter set
 - DSMPROF4, profile provided with 122
 - symbols provided with 23
 - tags contained in 35–115
 - tags, how described 35
 - using with other applications 7
 - GL (glossary list) tag description 60
 - glossary list, specifying 60
 - GML
 - See also* Generalized Markup Language tags
 - contained in starter set 35–115
 - definition of 4
 - processing of by SCRIPT/VS 117, 137
 - graphic formatting 32
 - GT (glossary term) tag description 60

H

- H0-H6 (headings 0-6) tag descriptions 62
- HDREF (heading reference) tag
 - description 65
- heading reference (HDREF tag) 65
- heading segments 15
- headings
 - referencing 65
 - specifying 62
- headings (H0-H6 tags) 62
- highlighted phrases (HP0-HP3 tags) 66
- highlighting text
 - as basic document elements 19
 - how to specify 66
- HP0-HP3 (highlighted phrases 0-3) tag description 66

hyphenation x

I

- I1-I3 (index entries 1-3) tag description 68
- IH1-IH3 (index headers 1-3) tag
 - description 71
- image object content architecture (IOCA) ix
- images 32
- imbedding files 27
- including overlays ix
- including page overlays (images) 30
- including page segments (images) 30
- including PostScript images 32
- index
 - as a basic document element 20
 - entries (I1-I3 tags) 68
 - entry reference (IREF tag) 75
 - headers (IH1-IH3 tags) 71
 - sorting entries 20
- INDEX option 120
- index sorting sequence 20
- INDEX tag description 74
- indicating revisions 28
- interim processing 136
- IREF (index entry reference) tag
 - description 75
- italic
 - highlighted phrases 66

L

- LaserPrinter 4028 support ix
- LI (list item) tag description 78
- LIB(name) option 120
- library guide
 - for DCF publications 158
- LIREF (list item reference) tag
 - description 77
- list item (LIREF tag) 77
- list of illustrations (FIGLIST tag) 52
- list of tables (TLIST tag) 108
- list tags (SL, OL, UL) 78
- lists
 - as basic document elements 16
 - definition 44
 - glossary 60
 - nesting 16
 - ordered 78
 - simple 78
 - unordered 78
- logical device, specifying 124

logical output devices
 table of 130
long quotation (LQ tag) 80
LP (list part) tag description 45, 78
LQ (long quotation) tag description 80

M

Macros, Control Words and 26
marking up documents 11
MARKUP 157
markup errors 137
markup/content separator, definition of 21
memo tags x
MESSAGE option 126
MINDEPTH on RDEF tag 90
modifying processing results 33
module renaming x
MVS and VM extended architecture ix

N

nesting lists 16
NOSEGLIB option 120
NOTE tag description 81

O

ODF 157
Office Document Feature (ODF) 157
OfficeVision 157
OL (ordered list) tag description 78
online help ix
online help for error messages ix
optional features
 Office Document Feature (ODF) 157
 SCRIPT Mathematical Formula Formatter (SMFF) 157
options
 BIND 118
 CHARS 118
 CONTINUE 126
 DEVICE 124, 131
 FILE 125
 FONTLIB 118
 FPASSES 120
 INDEX 120
 LIB(name) 120
 MESSAGE 126
 NOSEGLIB 120
 OPTIONS 121
 PAGE 121

options (*continued*)

PRINT 125
PROFILE 122
PSOUT 122
SEGLIB 122
SPELLCHK 124
SYSVAR 124
TERM 126
TWOPASS 124
OPTIONS option 121
output comment control word x
output destination, specifying 124
overhead transparency tags x
overlay include 30
override default fonts 127

P

P (paragraph) tag description 82
page number symbol limit x
PAGE option 121
page overlays 30
 with page devices 30
page printer font requirements 126
page printer fonts 126
page segment enhancements ix
page segments 30
 default libraries 123
 page segment libraries 122
 with page devices 30
paragraph (P tag) 82
paragraph unit, contents of 11
PC (paragraph continuation) tag description 82
phrases used, descriptions of 3
PostScript
 default file name, file type, file mode for 125
 default font libraries for 119
 images 32
 including PostScript images 32
PostScript fonts 127
PREFACE tag description 84
PRINT option ix, 125
Print Services Facility (PSF) 157
process-specific control (PSC tag) 85
processing results, modifying 33
products, related
 See related products
PROFILE option 122
profile, DSMPROF4
PSC (process-specific control) tag description 85
PSC element
 definition of 32

PSC element (*continued*)
 using 32
 PSF 157
 PSOUT option 122
 publication, related (see related publications)
 publications
 for DCF 159
 publications, referencing 43

Q

Q (quoted phrase) tag description 87

R

RDEF (row definition) tag description 88
 referencing
 figures 53
 footnotes 56
 headings 65
 index entries 75
 list items 77
 publications 43
 related products
 BookManager BUILD and READ 157
 BookMaster 157
 MARKUP 157
 OfficeVision 157
 Print Services Facility (PSF) 157
 SGML TextWrite 157
 SGML Translator DCF edition 157
 TextTagger ProcessMaster edition 157
 TextTagger Workstation edition 157
 related publications 157
 descriptions 160
 Release 4.0 changes
 font library index program (FLIP) enhance-
 ment x
 hyphenation x
 including overlays ix
 LANGUAGE attribute on the :GDOC
 tag x
 LaserPrinter 4028 support ix
 memo tags x
 module renaming x
 MVS and VM extended architecture ix
 online help ix
 online help for error messages ix
 output comment control word x
 overhead transparency tags x
 page number symbol limit x
 page segment enhancements
 ABSOLUTE parameter ix
 image object content architecture
 (IOCA) ix

Release 4.0 changes (*continued*)
 PRINT option ix
 removal of file size limitation ix
 removal of font character set size
 limitation ix
 revision character alignment x
 revision code font x
 schedule tags x
 separation masters ix
 shading ix, 92
 table storage relief x, 27
 TSO changes x
 removal of file size limitation ix
 removal of font character set size
 limitation ix
 revision character alignment x
 revision code font x
 revision codes 28
 ROTATE on RDEF tag 90
 row definition (RDEF) tag 88
 ROW tag description 95

S

schedule tags x
 SCRIPT command
 CHARS option 127
 issuing
 in the ATMS-III environment 132
 in the CMS environment 130
 in the TSO environment 131
 SCRIPT Mathematical Formula Formatter
 (SMFF) program 157
 SCRIPT/VS
 control words 26
 how it interprets GML 7
 logical devices 130
 macros 26
 processing text with 5
 symbols
 defining 27
 definition of 23
 provided with starter set 23
 SEGLIB option 122
 segment include 30
 self-study courses 159
 semi-colon symbol 24
 separation masters ix
 SGML TextWrite 157
 SGML Translator DCF edition 157
 Shading 92
 SL (simple list) tag description 78
 SMFF 157
 source document
 definition of 4

source document (*continued*)

management 26

space units 22

SPELLCHK option 124

structural elements

description of 14

listing of 14

Summary of Amendments ix

symbols

See also SCRIPT/VS, symbols

& 24

&date. 24

&gml. 24

&gml. instead of colon 24

&rbl. 24

&semi. 24

&time. 24

SYSVAR option 124

D (duplex) 133

H (head numbering) 133

P (process) 134

R (read) 134

S (style) 134

T (title page printing) 134

W (write) 135

X (cross-referencing) 135

T

table

caption (TCAP tag) 98

cell (C) tag 95

description (TDESC tag) 98

footer (TFT tag) 96

header (THD tag) 97

list (TLIST tag) 108

note (TNOTE tag) 109

reference (TREF tag) 110

rotate

on RDEF tag 88

row (ROW tag) 95

row definition (RDEF tag) 88

table header (THD tag) 97

Table Note (TNOTE tag) 109

table of contents (TOC tag) 114

table storage relief 27

TABLE tag description 93

tables, referencing 110

tag set for memos x

tag set for overhead transparencies x

tag set for schedules x

tags

ABSTRACT 37

ADDRESS 38

ALINE (address line) 38

tags (*continued*)

APPENDIX 39

AUTHOR 111

BACKM (back matter) 41

BODY 42

C (cell) 95

CIT (title citation) 43

DATE (document date) 111

DD (definition description) 45

DDHD (definition description heading) 45

DL (definition list) 44

DOCNUM (document number) 111

DT (definition term) 45

DTHD (definition term heading) 44

FIG (figure) 48

FIGCAP (figure caption) 48

FIGDESC (figure description) 48

FIGLIST (list of illustrations) 52

FIGREF (figure reference) 53

FN (footnote) 54

FNREF (footnote reference) 56

FRONTM (front matter) 57

GD (glossary definition) 60

GDOC (general document) 58

GL (glossary list) 60

GT (glossary term) 60

H0-H6 (headings 0-6) 62

HDREF (heading reference) 65

HP0-HP3 (highlighted phrases 0-3) 66

I1-I3 (index entries 1-3) 68

IH1-IH3 (index headers 1-3) 71

INDEX 74

IREF (index entry reference) 75

LI (list item) 78

LIREF (list item reference) 77

LP (list part) 45, 60, 78

LQ (long quotation) 80

NOTE 81

OL (ordered list) 78

P (paragraph) 82

PC (paragraph continuation) 82

PREFACE 84

PSC (process-specific control) 85

Q (quoted phrase) 87

RDEF (row definition) 88

ROW 95

SL (simple list) 78

TABLE 93

TCAP (table caption) 98

TDESC (table description) 98

TFT (table footer) 96

THD (table header) 97

TITLE (document title) 111

TITLEP (title page) 111

TLIST (list of tables) 108

TNOTE (table note) 109

tags (*continued*)

TOC (table of contents) 114
TREF (table reference) 110
UL (unordered list) 78
XMP (example) 115
TCAP (table caption) tag description 98
TDESC (table description) tag description 98
TERM option 126
text
 entry 22
 highlighting 22
 processing with SCRIPT/VS 5
TextTagger ProcessMaster edition 157
TextTagger Workstation edition 157
TFT (table footer) 96
THD (table header) tag description 97
time symbol 24
TITLE (document title) tag description 111
title citation (CIT tag) 43
title page (TITLEP tag) 111
TITLEP (title page) tag description 111
TLIST (list of tables) tag description 108
TNOTE (table note) tag description 109
TOC (table of contents) tag description 114
training courses 159
TREF (table reference) tag description 110
TSO changes x
TWOPASS option 124

X

XMP (example) tag description 115

U

UL (unordered list) tag description 78
units.measurement units
 See space
units.units of measure
 See space

V

VALIGN on RDEF tag 90
variable text
 reserving space for 29
VM and MVS extended architecture ix

W

writing comments 28

Reader's Comments

General Markup Language Starter Set Reference

Publication No. SH20-9187-06

Use this form to provide comments about this publication, its organization, or subject matter. Understand that IBM may use the information any way it believes appropriate, without incurring any obligation to you. Your comments will be sent to the author's department for the appropriate action. Comments may be written in your language.

Note: IBM publications are not stocked at the location to which this form is addressed. Direct requests for publications or for assistance in using your IBM system, to your IBM representative or local IBM branch office.

- | | Yes | No |
|---|--------------------------|--------------------------|
| • Does the publication meet your needs? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Did you find the information: | | |
| Accurate? | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to read and understand? | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to retrieve? | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use? | <input type="checkbox"/> | <input type="checkbox"/> |
| Legible? | <input type="checkbox"/> | <input type="checkbox"/> |
| Complete? | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated? | <input type="checkbox"/> | <input type="checkbox"/> |
| Written for your technical level? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Do you use this publication: | | |
| As an introduction to the subject? | <input type="checkbox"/> | <input type="checkbox"/> |
| As a reference manual? | <input type="checkbox"/> | <input type="checkbox"/> |
| As an instructor in class? | <input type="checkbox"/> | <input type="checkbox"/> |
| As a student in class? | <input type="checkbox"/> | <input type="checkbox"/> |
| • What is your occupation? | | |

Thank you for your input and cooperation.

Note: If mailed in the U.S.A., no postage stamp is necessary. For residents outside the U.S.A., your local IBM office or representative will forward your comments.

Comments:

Name

Address

Company or Organization

Phone No.

Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 588
P.O. Box 1900
Boulder, CO 80301-9191



Fold and Tape

Please do not staple

Fold and Tape

Reader's Comments

General Markup Language Starter Set Reference

Publication No. SH20-9187-06

Use this form to provide comments about this publication, its organization, or subject matter. Understand that IBM may use the information any way it believes appropriate, without incurring any obligation to you. Your comments will be sent to the author's department for the appropriate action. Comments may be written in your language.

Note: IBM publications are not stocked at the location to which this form is addressed. Direct requests for publications or for assistance in using your IBM system, to your IBM representative or local IBM branch office.

- | | Yes | No |
|---|--------------------------|--------------------------|
| • Does the publication meet your needs? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Did you find the information: | | |
| Accurate? | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to read and understand? | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to retrieve? | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use? | <input type="checkbox"/> | <input type="checkbox"/> |
| Legible? | <input type="checkbox"/> | <input type="checkbox"/> |
| Complete? | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated? | <input type="checkbox"/> | <input type="checkbox"/> |
| Written for your technical level? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Do you use this publication: | | |
| As an introduction to the subject? | <input type="checkbox"/> | <input type="checkbox"/> |
| As a reference manual? | <input type="checkbox"/> | <input type="checkbox"/> |
| As an instructor in class? | <input type="checkbox"/> | <input type="checkbox"/> |
| As a student in class? | <input type="checkbox"/> | <input type="checkbox"/> |
| • What is your occupation? | | |

Thank you for your input and cooperation.

Note: If mailed in the U.S.A., no postage stamp is necessary. For residents outside the U.S.A., your local IBM office or representative will forward your comments.

Comments:

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 588
P.O. Box 1900
Boulder, CO 80301-9191



Fold and Tape

Please do not staple

Fold and Tape

Reader's Comments

General Markup Language Starter Set Reference

Publication No. SH20-9187-06

Use this form to provide comments about this publication, its organization, or subject matter. Understand that IBM may use the information any way it believes appropriate, without incurring any obligation to you. Your comments will be sent to the author's department for the appropriate action. Comments may be written in your language.

Note: IBM publications are not stocked at the location to which this form is addressed. Direct requests for publications or for assistance in using your IBM system, to your IBM representative or local IBM branch office.

- | | Yes | No |
|---|--------------------------|--------------------------|
| • Does the publication meet your needs? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Did you find the information: | | |
| Accurate? | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to read and understand? | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to retrieve? | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use? | <input type="checkbox"/> | <input type="checkbox"/> |
| Legible? | <input type="checkbox"/> | <input type="checkbox"/> |
| Complete? | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated? | <input type="checkbox"/> | <input type="checkbox"/> |
| Written for your technical level? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Do you use this publication: | | |
| As an introduction to the subject? | <input type="checkbox"/> | <input type="checkbox"/> |
| As a reference manual? | <input type="checkbox"/> | <input type="checkbox"/> |
| As an instructor in class? | <input type="checkbox"/> | <input type="checkbox"/> |
| As a student in class? | <input type="checkbox"/> | <input type="checkbox"/> |
| • What is your occupation? | | |

Thank you for your input and cooperation.

Note: If mailed in the U.S.A., no postage stamp is necessary. For residents outside the U.S.A., your local IBM office or representative will forward your comments.

Comments:

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 588
P.O. Box 1900
Boulder, CO 80301-9191

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



Fold and Tape

Please do not staple

Fold and Tape